Induction and Skolemization in saturation theorem proving

Stefan Hetzl [*], Jannik Vierling

Vienna University of Technology, Institute of Discrete Mathematics and Geometry, Austria

A R T I C L E   I N F O

A B S T R A C T

We consider a typical integration of induction in saturation-based theorem provers
and investigate the effects of Skolem symbols occurring in the induction formulas.
In a practically relevant setting we establish a Skolem-free characterization of
refutation in saturation-based proof systems with induction. Finally, we use this
characterization to obtain unprovability results for a concrete saturation-based
induction prover.

© 2022 The Author(s). Published by Elsevier B.V. This is an open access article
under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Automated inductive theorem proving (AITP) is a branch of automated deduction that aims at automating the process of finding proofs that involve mathematical induction. In first-order automated theorem proving (ATP) we try to establish validity whereas in automated inductive theorem proving (AITP) one is usually interested to prove that a formula is true in the standard model of some inductive type, such as natural numbers, lists, trees, etc. By Gödel's incompleteness theorems, truth in the standard model is in general not semi-decidable (even worse, it is in general not even arithmetically definable). Hence, for AITP there is a lot more freedom in the choice of proof systems, than there is for ATP. In practice we see methods that make use of typical first-order induction schemata, Hilbert-style induction rules (for example [31,29]), and even more exotic cyclic calculi (see [9,4]) that can exceed the power of the first-order induction schema [11,12].

---

* Corresponding author.
E-mail addresses: stefan.hetzl@tuwien.ac.at (S. Hetzl), jannik.vierling@tuwien.ac.at (J. Vierling).

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. Besides the applications in software engineering, AITP methods have applications in the formalization of mathematics. For instance, AITP methods can be employed by proof assistants to explore a theory in order to provide useful lemmas [28], [26].

A wide variety of methods for automated inductive theorem proving have been developed: there are methods based on recursion analysis [8,41,13], proof by consistency [17], rippling [10], cyclic proofs [4], extensions of saturation-based provers [5,31,29,18,19,22,36,23,46], tree grammar provers [21], theory exploration based provers [15], rewriting induction [34], encoding [38], extensions of SMT solvers [35]. Many methods integrate the induction mechanism more or less tightly within a proof system that is well-suited for automation. Therefore, these methods exist mainly at lower levels of abstraction, often close to an actual implementation. Such methods are traditionally evaluated empirically on a set of benchmark problems such as the one described by Claessen et al. [16]. Formal explanations backing the observations obtained by the empirical evaluation are still rare. As of now, it is difficult to classify methods according to their strength and to give theoretical explanations of an empirically observed failure of a given method in a particular context.

The work in this article is part of a research program that aims at analyzing methods for AITP by applying techniques and results from mathematical logic. The purpose of this is twofold. Firstly, formal analyses allow us to complement and to explain the empirical knowledge obtained by the practical evaluations of AITP methods. Secondly, the analyses carried out during this program will inevitably lead to a development of the logical foundations of automated inductive theorem proving. In particular, we believe that practically relevant negative results are especially valuable in revealing the features a method is lacking. Thus, negative results may drive the development of new methods. Moreover, we believe that this research program will strengthen the link between the research in automated inductive theorem proving and mathematical logic, and therefore, may lead to cross-fertilization by providing interesting theoretical techniques from mathematical logic and new problems for mathematical logic.

As part of this research program Hetzl and Wong [25] have given some observations on the logical foundations of inductive theorem proving. Vierling [43] has analyzed the n-clause calculus [31,29] resulting in an estimate of the strength of this calculus. Building on this analysis Hetzl and Vierling [24] have further abstracted the n-clause calculus and situated this calculus with respect to some logical theories. The authors are currently also working on an unprovability result for the n-clause calculus.

The subject of AITP has recently increasingly focused on integrating mathematical induction in saturation-based theorem provers [31,29,18,19,46,22,36,23]. In this article we propose abstractions of these systems and investigate how Skolemization interferes with induction in such a system. In a fairly general yet practically relevant setting we are able to show that Skolem symbols take the role of induction parameters. We use this insight to provide unprovability results for a family of methods using induction for quantifier-free formulas. This allows us in particular to obtain unprovability results for the concrete method described in [36,23].

In this article we will provide a unified view of a commonly used strategy to integrate induction into saturation-based theorem proving and concentrate on the role of Skolemization in these systems. To our knowledge the interaction between induction and Skolemization has not been investigated in the related literature. Section 2 introduces all the necessary notations related to our logical formalism, our presentation of Skolemization, and the arithmetic theories used in this article. We will give a precise presentation of Skolemization, that imposes a concrete naming schema which will be particularly useful in dealing with the languages generated by saturation systems. In Section 3 we give an abstract description of saturation-based proof systems and describe abstractly a common strategy to integrate induction in such systems. We furthermore present a restriction of this system that generalizes a way to handle induction found in

most practical saturation systems with induction. Section 4 gives a very clear characterization of refutation in saturation systems with an unrestricted induction rule (see Theorem 4.11) and analyzes the effects of Skolemization on the induction. In Section 5 we analyze the effect of Skolemization in syntactically restricted systems that are closer to the practical methods. This section culminates in a Skolem-free characterization of these systems (see Theorem 5.23). Finally in Section 6 we make use of the results from Section 5 to provide practically relevant unprovability results for a family of methods using quantifier-free induction formulas (see Theorem 6.6) and apply this result to the concrete method presented in [36,23].

2. Preliminary definitions

In this section we settle the details of the logical formalism that we use throughout the article. For the sake of clarity we try to adhere as much as possible to standard terminology, but we introduce some non-standard notations where it is beneficial for the presentation. In Section 2.1 we describe our logical formalism and the related notations such as clauses. Section 2.2 introduces some definitions and well-known results related to Skolemization and in particular the naming schema for Skolem symbols that we adopt in this article. Finally, in Section 2.3 we recall some notions of formal arithmetic and introduce a particular theory of formal arithmetic that will be of use at various occasions.

2.1. Formulas, theories, and clauses

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. In the context of automated theorem proving it is common to work in a many-sorted setting, but in order to keep the presentation simple we only use one sort. All our definitions and results easily generalize to the many-sorted case. A first-order language L is a countable set of function symbols and predicate symbols with their respective arities. Let σ be a (function or predicate) symbol, then we write σ/n to denote that σ has arity $n \in \mathbb{N}$. Terms are constructed from function symbols and variables. Formulas are constructed as usual from atomic formulas, the connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, and the quantifiers $\exists$ and $\forall$. In order to save some parentheses we assume the following order of precedence for the propositional connectives: $\neg$, $\vee$, $\wedge$, $\rightarrow$. By $\mathcal{F}(L)$ we denote the set of L formulas. 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. By $(\exists! y)\varphi(\vec{x}, y)$ we abbreviate the formula

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

In this article we are 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 2.2).

Definition 2.1 *(Theories).* Let L be a first-order language, then a first-order L theory T is a set of L sentences called the axioms of T.

For the sake of legibility we often present the axioms of a theory as a list of formulas with free variables, with the intended meaning that these formulas are universally closed. By $L(T)$ we denote the language of the theory T. When no confusion arises we sometimes write T in places where $L(T)$ is expected.

Definition 2.2 *(Provability).* Let φ be a sentence and T a theory, then we write $T \vdash \varphi$ to denote that φ is provable in first-order logic from the axioms of T. Let Γ be a set of sentences, then we write $T \vdash \Gamma$ to denote that $T \vdash \varphi$ for all sentences $\varphi \in \Gamma$. Let T_1 and T_2 be theories, then we write $T_1 \equiv T_2$ if $T_1 \vdash T_2$ and $T_2 \vdash T_1$.

Let $\varphi(\vec{x})$ be a formula and T a theory, then in order to ease the notation we will sometimes write $T \vdash \varphi(\vec{x})$ in place of $T \vdash (\forall \vec{x})\varphi(\vec{x})$.

Definition 2.3 *(Conservativity).* Let T_1 and T_2 be theories, and Γ a set of formulas. We say that T_1 is Γ-conservative over T_2 (in symbols $T_1 \sqsubseteq_\Gamma T_2$), if, for all $\varphi \in \Gamma$, $T_1 \vdash \varphi$ implies $T_2 \vdash \varphi$. We write $T_1 \equiv_\Gamma T_2$ if $T_1 \sqsubseteq_\Gamma T_2$ and $T_1 \sqsupseteq_\Gamma T_2$. If $\Gamma = \mathcal{F}(L)$ for some first-order language L, then we may simply write $T_1 \sqsubseteq_L T_2$ for $T_1 \sqsubseteq_{\mathcal{F}(L)} T_2$.

Automated theorem provers—in particular saturation systems—usually do not operate directly on formulas but instead operate on clauses and clause sets (see Section 3).

Definition 2.4 *(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. An L clause set is a set of clauses. By $\square$ we denote the empty clause. Let C and D be clauses, then we write $C \vee D$ for the union of the clauses C and D. Let $\mathcal{C}$ be a clause set and D a clause, then we write $\mathcal{C} \vee D$ to denote the clause set $\{C \vee D \mid C \in \mathcal{C}\}$. Furthermore, we write $L(\mathcal{C})$ to denote the language of $\mathcal{C}$, that is, the set of non-logical symbols that occur in clauses of $\mathcal{C}$.

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.

We will now recall basic some model-theoretic concepts and notations. Let L be a language, then an L structure is a pair $M = (D, I)$, where D is a non-empty set and I is an interpretation. The interpretation I is a function that assigns to each symbol $\sigma/k \in L$ an interpretation σ^I such that if σ is a predicate symbol, then $\sigma^I \subseteq D^k$ and if σ is a function symbol, then $\sigma^I : D^k \to D$. Let $\varphi(x_1, \ldots, x_n)$ be an L formula and $d_1, \ldots, d_n \in D$, then we write $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models \varphi$ if φ is true in M under the variable assignment that assigns d_i to x_i for $i = 1, \ldots, n$.

Definition 2.5 *(Notation).* Let L be a language, $M = (D, I)$ an L structure, then we define $|M| = D$. Moreover, we sometimes write $d \in M$ if $d \in D$ and for a symbol $\sigma \in L$, we also denote its interpretation σ^I in M by σ^M. Let $\varphi(x_1, \ldots, x_n)$ be an L formula and $d_1, \ldots, d_n \in D^{|\vec{x}|}$, then we write $M \models \varphi(d_1, \ldots, d_n)$ if $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models \varphi$. Furthermore, we write $M \models \varphi$, if $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models \varphi$, for all $d_1, \ldots, d_n \in M$. Similarly, we write $M \models C$ for an L clause C with free variables $x_1, \ldots, x_n$, if $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models C$ for all $d_1, \ldots, d_n \in M$. Let Δ be a set of formulas and clauses, then we write $M \models \Delta$ if $M \models \delta$ for each $\delta \in \Delta$. We write $\Lambda \models \Delta$ if for every model M of Λ we have $M \models \Delta$.

Definition 2.6. Let L be a language and M a first-order structure, then we define

$$\mathrm{Th}(M) := \{\varphi \mid M \models \varphi, \varphi \text{ is an } L \text{ sentence}\}.$$

We are often interested in the formulas that have a certain structure.

Definition 2.7. 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 $(\exists \vec{x})\varphi(\vec{x}, \vec{y})$ $((\forall \vec{x})\varphi(\vec{x}, \vec{y}))$, where φ is $\forall_n$ ($\exists_n$) and $\vec{x}$ is a possibly empty vector of variables. Let L be a first-order language, then by $\mathrm{Literal}(L)$, $\mathrm{Open}(L)$, $\exists_n(L)$, and $\forall_n(L)$ we denote the set of literals, open formulas, $\exists_n$ formulas, and $\forall_n$ formulas of the language L. We say that a theory is $\forall_n$ ($\exists_n$) if all of its axioms are $\forall_n$ ($\exists_n$).

As mentioned above, automated theorem provers often work on sets of clauses, rather than formulas. Hence, it is necessary to discuss how formulas are associated with clause sets. In the following definition we fix one such translation that we use throughout the article.

Definition 2.8. By *CNF* we denote a fixed function that assigns to any $\forall_1$ sentence φ, a clause set C_φ such that $L(\varphi) = L(C_\varphi)$ and φ and C_φ are logically equivalent. 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. Since this article focuses on the interaction of induction and Skolemization, we choose to exclude conjunctive normal form translations that do not preserve the language. The question how these more advanced transformations interact with induction is clearly also important and should be investigated separately.

2.2. Skolemization

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 [7], 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.

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

Definition 2.9. Let L be a first-order language, then we define

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

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

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

Definition 2.10. We define the functions $sk^\forall, sk^\exists : \mathcal{F}(sk^\omega(L)) \to \mathcal{F}(sk^\omega(L))$ mutually inductively as follows

$$sk^Q(P(\vec{t})) := P(\vec{t}),$$
$$sk^Q(A \wedge B) := sk^Q(A) \wedge sk^Q(B),$$
$$sk^Q(A \vee B) := sk^Q(A) \vee sk^Q(B),$$
$$sk^Q(\neg A) := \neg sk^{\overline{Q}}(A),$$
$$sk^Q((Qx)A(x, \vec{y})) := sk^Q(A(\mathfrak{s}_{(Qx)A(x,\vec{y})}(\vec{y}), \vec{y})), \qquad (*)$$
$$sk^Q((\overline{Q}x)A) := (\overline{Q}x)sk^Q(A),$$

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 Γ be a set of formulas, then we define $sk^Q(\Gamma) := \{sk^Q(\varphi) \mid \varphi \in \Gamma\}$.

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 2.11. 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 2.13.

The following property of Skolemization is well-known.

Proposition 2.12. *Let L be first-order language and φ an $sk^\omega(L)$ 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. These axioms essentially correspond to the existential Skolem form of the logical axioms $\varphi \to \varphi$.

Definition 2.13. Let L be a first-order language, and $\varphi(x,\vec{y})$ an $sk^\omega(L)$ formula, then we define

$$
\begin{aligned}
\mathrm{SA}_x^\exists\varphi &:= (\exists x)\varphi \to \varphi(\mathfrak{s}_{(\exists x)\varphi}(\vec{y}),\vec{y}), \\
\mathrm{SA}_x^\forall\varphi &:= \varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}),\vec{y}) \to (\forall x)\varphi.
\end{aligned}
$$

We define $L\text{-SA} := \{(\forall\vec{y})\mathrm{SA}_x^Q\varphi \mid Q \in \{\forall,\exists\}, \mathfrak{s}_{(Qx)\varphi(x,\vec{y})} \in sk^\omega(L)\}$.

The Skolem axioms allow us to also obtain the converse of Proposition 2.12.

Proposition 2.14. *Let L be a first-order language, φ an $sk^\omega(L)$ formula, and $Q \in \{\forall,\exists\}$. Then we have $L\text{-SA} \vdash \varphi \leftrightarrow sk^Q(\varphi)$.*

Proof. Straightforward. $\square$

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

Proposition 2.15. *Let L be a Skolem-free first-order language and T be an L theory, then $L\text{-SA} + T \equiv_L T$.*

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 2.16. *Let L be a Skolem-free language and T be an L theory, then*

$$sk^\exists(T) \equiv_L T.$$

Proof. The direction $sk^\exists(T) \sqsubseteq_L T$ is an immediate consequence of Proposition 2.12. For the other direction we have $T \equiv_L^{\text{Prop. 2.15}} L\text{-SA} + T \equiv^{\text{Prop. 2.14}} L\text{-SA} + sk^\exists(T)$. Hence $T \equiv_L sk^\exists(T)$. $\quad\square$

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

Corollary 2.17. *Let 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. Induction and arithmetic

We conclude the preliminary definitions with the definition of some notions related to formal arithmetic. Let us start by discussing the setting for induction that we use in this article. In automated inductive theorem proving it is customary to work with various inductively defined objects such as the natural numbers, lists, trees, and mutually recursive constructions. Typically inductive theorem proving concentrates on a multi-sorted setting where a subset of the sorts is interpreted as the term algebra constructed over a set of function symbols, called the constructors. Such a construction, while of great practical relevance, incurs significant notational complexity. Therefore, in order to avoid overloading the presentation, we restrict our setting to the natural numbers. However, we expect that our results straightforwardly carry over to the more general case mentioned above, because the structure of the induction axiom remains essentially the same.

Definition 2.18. By $0/0$ and $s/1$ we denote the function symbols representing the natural number 0 and the successor function, respectively. Moreover, we let $L_0 := \{0/0, s/1\}$.

We can now define induction axioms and the first-order structural induction schema.

Definition 2.19. Let L be a language, and $\varphi(x, \vec{z})$ be an L formula, then the $L \cup L_0$ formula $\tilde{I}_x\varphi$ is given by

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

We refer to the variable x as the induction variable and to the variables $\vec{z}$ as the induction parameters. Moreover we define the induction axiom $I_x\varphi$ by $I_x\varphi := (\forall \vec{z})\tilde{I}_x\varphi$. Let Γ be a set of L formulas, then the set of $L \cup L_0$ sentences $\Gamma\text{-IND}$ is given by $\{I_x\gamma \mid \gamma(x, \vec{z}) \in \Gamma\}$.

By an arithmetical language we understand a first-order language containing the symbols $0/0$, $s/1$, and possibly some symbols representing primitive recursive functions. In the following definition we recall some standard terminology for arithmetic.

Definition 2.20. Let L be an arithmetical language. By $\mathbb{N}_L$ the structure whose domain is the set of natural numbers and that interprets the non-logical symbols of L in the natural way. An arithmetical theory is a theory over an arithmetical language. Let T be an L theory. We say that the theory T is sound if $\mathbb{N}_L \models T$. Furthermore, we say that T is $\exists_1$-complete if $\mathbb{N}_L \models \varphi$ implies $T \vdash \varphi$ for all $\exists_1$ L sentences.

We conclude this section by describing the setting of linear arithmetic that will in particular serve us in Section 6.2 for obtaining unprovability results for the methods [36,23]. The language of linear arithmetic contains besides $0/0$ and $s/1$ only the function symbols $p/1$ and $+/2$ as infix symbol, where p denotes the predecessor function and $+$ denotes the addition. Clearly, the setting of linear arithmetic is closely related to Presburger arithmetic. However, we are not interested in the theory of the standard interpretation, but rather in its subtheories such the ones that were already studied by Shoenfield [40]. This setting of linear arithmetic turns out to be quite useful in the analysis of methods for automated inductive theorem proving, because on the one hand it is simple enough to still allow for straightforward model-theoretic constructions, yet it is complex enough to provide interesting independence results.

Let us fix some notational conventions. Let $m \in \mathbb{N}$ and t be a term, then by $m \cdot t$ we denote the term $t + (t + \cdots + (t + t) \cdots)$. Let f be a unary function symbol, then $f^m(t)$ stands for $f(\cdots f(t) \cdots)$. By $\overline{m}$ we denote the term $s^m(0)$. Our base theory for linear arithmetic is defined as follows.

Definition 2.21. By $\mathcal{T}$ we denote the theory axiomatized by the universal closure of the following formulas

$$0 \neq s(x), \tag{A1}$$

$$p(0) = 0, \tag{A2}$$

$$p(s(x)) = x, \tag{A3}$$

$$x + 0 = x, \tag{A4}$$

$$x + s(y) = s(x + y), \tag{A5}$$

We conclude with two basic observations about the theory $\mathcal{T}$. We shall make use of these observations at several occasions and will for the sake of readability not mention them explicitly every time.

Lemma 2.22. $\mathcal{T} \vdash s(x) = s(y) \to x = y$.

Proof. Use (A3). $\square$

Proposition 2.23. $\mathcal{T}$ *is sound and* $\exists_1$*-complete.*

Proof. The soundness part is obvious. For the $\exists_1$-completeness observe that $\mathcal{T}$ decides ground formulas. $\square$

3. Saturation-based systems and induction

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 [5]. Another, currently more popular, way is to integrate the induction mechanism more tightly into the saturation system as some form of inference rule [31,29], [36,23], [18,19], [46], [22]. In this section we give an abstract framework for AITP methods integrating induction in saturation-proof systems in terms of a general induction rule. This framework will allow us to investigate in Sections 4 and 5 the role of Skolem symbols in these systems. In Section 6 we show that the methods described in [36,23] fit into our framework. In Section 3.1 we define saturation systems abstractly and introduce some related notions. After that, Section 3.2 introduces the notion of induction rule as a general way to integrate induction into a saturation system and presents a practically relevant specialization of this induction rule.

3.1. Saturation-based proof systems

Saturation is a technique of automated theorem proving that consists of computing the closure of a set of formulas or clauses 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 less structure and are therefore better suited for automated proof search.

In what follows we concentrate on the 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 provable in 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.

Practical 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 that generate new clauses. In particular, our notion of saturation system does not have any notion of redundancy mechanisms such as simplification rules and deletion rules. Since this article is mostly about upper bounds on the logical strength of AITP methods, the assumption that clauses are never deleted is unproblematic.

Definition 3.1 *(Saturation systems).* A saturation system $\mathcal{S}$ is a set of inference rules of the form

$$\frac{\mathcal{C}}{\mathcal{D}},$$

also written as $\mathcal{C}/\mathcal{D}$ where $\mathcal{C}$ is a set of clauses $\mathcal{D}$ is a finite set of clauses. 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 $\mathcal{S}_2$.

Informally, an inference rule $\mathcal{C}/\mathcal{D}$ indicates that if the system is in the "state" $\mathcal{C}$, then the system changes into the "state" $\mathcal{C} \cup \mathcal{D}$. The reason why we consider inference rules of this form is that they allow us to keep track of global properties of the prover such as for example the language of the currently derived clauses. Observe that our notion of inference rules is very general since $\mathcal{C}$ may be infinite. Hence we could formulate an ω-rule for saturation systems. However, we will only work with inference rules that operate with the language of $\mathcal{C}$ and a finite set of clauses $\mathcal{C}_0 \subseteq \mathcal{C}$.

Example 3.2. The resolution rule can be presented as follows:

$$\frac{\{l \vee C\} \cup \{m \vee D\} \cup \mathcal{C}}{\{(C \vee D)\mu\}} \text{ Res,}$$

where $\mathcal{C}$ is a clause set, C and D are clauses, and μ is the most general unifier of the literals l and $\overline{m}$.

Definition 3.3 *(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 $\mathcal{D}_0, \ldots, \mathcal{D}_n$ such that $\mathcal{C}_0 = \mathcal{D}_0$ and $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \mathcal{B}_i$ such that $\mathcal{D}_i/\mathcal{B}_i$ is an inference rule of $\mathcal{S}$ for $0 \leq i < n$. We say that a clause C is derivable from $\mathcal{C}_0$ in $\mathcal{S}$ if there exists a deduction $\mathcal{D}_0, \ldots, \mathcal{D}_n$ such that $C \in \mathcal{D}_n$. A deduction $\mathcal{D}_0, \ldots, \mathcal{D}_n$ is called a refutation if $\square \in \mathcal{D}_n$.

Since we are usually interested in extending saturation systems for pure first-order logic by inference rules for induction we need to introduce the notion of soundness and refutational completeness.

Definition 3.4. 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 $L(C) \subseteq L(\mathcal{C}_0)$ 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.

3.2. Induction rules

Typically induction is integrated in a saturation prover by a mechanism, 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 [31,29,36,23,18,46]. 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 axiom is added to the search space. For instance, Kersani and Peltier's method [31,29] 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. We abstract the induction mechanisms of the aforementioned methods by the following induction rule.

Definition 3.5. The induction rule IND^R is given by

$$\frac{\mathcal{C}}{CNF(sk^\exists(I_x\varphi(x, \vec{z})))} \ \mathrm{IND}^R$$

where $\mathcal{C}$ is a set of clauses, $\varphi(x, \vec{z})$ is a $L(\mathcal{C})$ formula.

Despite being limited to natural numbers, the induction rule presented above is very general in the sense that it does not impose any restrictions on the complexity of the induction formulas. None of the methods known to us comes even close to making use of the full power offered by that rule. Nevertheless, it will serve us as a useful tool for theoretical analyses.

There is an important observation that we can make about this induction rule. 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^R permits Skolem symbols to appear in induction formulas. In other words, the induction IND^R iteratively extends the language of the induction formulas by Skolem symbols. Interestingly, a similar situation has been considered in the literature on mathematical logic [3]. In saturation systems for pure first-order logic, the role of Skolemization is clear: It allows us to obtain an equiconsistent formula without existential quantifiers (see Corollary 2.17). In saturation systems with the induction rule IND^R the role of Skolemization is not clear anymore, in the sense of Corollary 2.17. This raises the question how the extension of the language of induction formulas by Skolem symbols affects the power of the system. Also note that this feature is not artificial but actually appears in the concrete methods mentioned above.

We shall address this question in Section 4. In particular we will provide a logical characterization of refutability in a sound and complete saturation system extended by the induction rule IND^R in terms of a theory with an induction schema (see Theorem 4.11). As a corollary we obtain the soundness of the rule IND^R (see Corollary 4.12).

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

Example 3.6. 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 $\mathcal{S} + \mathrm{IND}^R$ of the clause set $\mathcal{C}_0$ given by

$$CNF(sk^{\exists}(\mathcal{T} + \neg(\forall x)(\forall y)x + y = y + x)).$$

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

$$\mathcal{C}_0 \models c_1 + c_2 \neq c_2 + c_1. \tag{1}$$

Let $\varphi_1(x) := (c_1 + x = x + c_1)$, then we may apply the induction rule IND^R to obtain the clause set $\mathcal{C}_1 := \mathcal{C}_0 \cup CNF(sk^{\exists}(I_x\varphi_1(x)))$. 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 (1) we have

$$\mathcal{C}_1 \models \neg\varphi_1(0) \vee \neg(\varphi_1(c_3) \to \varphi_1(s(c_3))). \tag{2}$$

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 $\varphi_2(x) := x = 0 + x$, then we apply the induction rule IND^R in order to obtain the clause set $\mathcal{C}_2 := \mathcal{C}_1 \cup CNF(sk^{\exists}(I_x\varphi_2))$. Let $sk^{\exists}(I_x\varphi_2) := (\varphi_2(0) \wedge (\varphi_2(c_4) \to \varphi_2(s(c_4)))) \to (\forall x)\varphi_2$, then by (2) we have

$$CNF(\mathcal{C}_2) \models \neg\varphi_2(0) \vee \neg(\varphi_2(c_4) \to \varphi_2(s(c_4))) \vee \neg(\varphi_1(c_3) \to \varphi_1(s(c_3))). \tag{3}$$

Now observe that $\mathcal{T} \models 0 = 0 + 0$ and $\mathcal{T} \models 0 + s(c_4) = s(0 + c_4)$. Hence, $\mathcal{T} \models c_4 = 0 + c_4 \to s(c_4) = s(0 + c_4)$, that is, $\mathcal{T} \models \varphi_2(c_4) \to \varphi_2(c_4)$ and $\mathcal{T} \models \varphi_2(0)$. Therefore, by (3) we obtain

$$\mathcal{C}_2 \models \neg(\varphi_1(c_3) \to \varphi_1(s(c_3))). \tag{4}$$

Recall that $\varphi_1(x) = (c_1 + x = x + c_1)$. Since $\mathcal{T} \models c_1 + s(x) = s(c_3 + x)$, we have by (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

$$\mathcal{C}_2 \models \neg\varphi_3(c_3). \tag{5}$$

Now we apply the induction rule IND^R 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 (5) we have

$$\mathcal{C}_3 \models \neg\varphi_3(0) \vee \neg(\varphi_3(c_5) \to \varphi_3(s(c_5)).) \tag{6}$$

Since $CNF(\mathcal{T}) \models s(c_3 + 0) = s(c_3) = s(c_3) + 0$, we have $\mathcal{C}_3 \models \varphi(0)$. Moreover, $CNF(\mathcal{T}) \models s(c_3 + s(c_5)) = s(s(c_3 + c_5))$, hence $CNF(\mathcal{T}) \models \varphi_3(c_5) \to \varphi_3(s(c_5))$. Hence, by (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^R used to obtain $\mathcal{C}_3$ with the $\mathcal{S}$ refutation of $\mathcal{C}_3$ we obtain a $\mathcal{S} + \text{IND}^R$ refutation of $\mathcal{C}_0$.

Analyzing the rule IND^R will give us some general insights about the role of Skolem symbols in saturation systems with induction, however in order to be more specific about particular methods we have to consider some restricted forms of this induction rule. We start by introducing some additional terminology. We call *initial Skolem symbols* those Skolem symbols that arise from the Skolemization of the input problem and *induction Skolem symbols* those Skolem symbols that are generated by an application of the induction rule.

Before we introduce a restriction of the induction rule that is of practical relevance we will discuss some remarkable design choices encountered in practical methods that we will incorporate into the induction rule:

- Syntactical restriction of induction formulas: The methods presented in [36,23] restrict induction formulas to literals, [31,29] restricts induction formulas to $\exists_1$ formulas, and [18,19] 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 [31,29] or by introducing nullary Skolem symbols only [36,23], [18,19]. 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 are 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 Skolem symbols that are not nullary. Hence variables may 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 3.7. Let Γ be a set of formulas, then the rule $\Gamma\text{-GIND}^R$ is given by

$$\frac{\mathcal{C}}{CNF(sk^{\exists}(I_x\varphi(x,\vec{t})))}\ \Gamma\text{-GIND}^R,$$

where $\mathcal{C}$ is a set of clauses, $\varphi(x,\vec{z}) \in \Gamma$, and $\vec{t}$ is a vector of ground $L(\mathcal{C})$ terms.

Remark 3.8. 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 [20], Skolemization without this restriction in simple type theory makes the axiom of choice derivable, hence this restriction has been introduced in [33]. This restriction is also used as an assumption for proving elementary deskolemization of proofs with cut in [6], [30].

Let us again consider an example to illustrate the rule.

Example 3.9. Consider the refutation carried out in Example 3.6. We have used the induction rule three times to derive the clause sets $CNF(sk^{\exists}(I_x c_1 + x = x + c_1))$, $CNF(sk^{\exists}(I_x x = 0 + x))$, and $CNF(I_x s(c_3 + x) = s(c_3) + x)$. All three induction formulas are equational atoms in which only nullary Skolem symbols appear. Hence the refutation outlined in Example 3.6 is also a refutation in $\mathcal{S}+\text{Eq}(\mathcal{T})\text{-GIND}^R$, where $\text{Eq}(L)$ denotes the set of equational atoms over the language L.

As with the rule IND^R we now have to ask the question how the system behaves. There are two major cases that we need to distinguish depending on whether the set of formulas Γ may contain initial Skolem symbols. By letting Γ be a set of Skolem-free formulas, we can restrict the occurrences of all Skolem symbols in the induction formulas. In Section 5 we mainly concentrate on this case and provide a characterization for the refutability in a sound and refutationally complete saturation system with the rule $\Gamma\text{-GIND}^R$, thus, settling the question. In practical systems the initial Skolem symbols usually can appear in the induction formulas without restriction, that is, these systems correspond to the case where the formulas in Γ may contain initial Skolem symbols. However, this case is actually part of a more general open problem concerning occurrences of Skolem symbols in axiom schemata, that we will not address in this article (see Remark 3.8). Nevertheless, we can handle the simple case when the initial Skolem symbols are nullary. We will mainly

deal with this case in Section 6 in order to provide an unprovability result for the methods described in [36] and [23].

4. Unrestricted induction and Skolemization

In the previous section we have abstractly described a common integration of induction into a saturation system via the induction rule IND^R. In this section we will first represent a sound and refutationally complete saturation system extended by the rule IND^R as a logical theory. After that we make use of this representation in order to investigate the interaction between Skolemization and the induction rule.

4.1. Representation as logical theory

A useful technique when analyzing AITP methods is to reduce the system to an "equivalent" logical theory. Alternatively, when such a theory cannot be found it is a good practice to approximate the system by a logical theory as closely as possible. The construction of that theory usually reveals the essential features of the method. Moreover, we can then make use of powerful techniques from mathematical logic in order to study the theory. In particular, we can compare methods in terms of their representative theories.

Definition 4.1. Let T be a theory, then we define the Skolem induction operator SI by

$$\text{SI}(T) \coloneqq T + sk^{\exists}(L(T)\text{-IND})$$

By $\text{SI}^i(T)$ we denote the i-fold iteration of SI on T. Finally, we define $\text{SI}^{\omega}(T) \coloneqq \bigcup_{i<\omega} \text{SI}^i(T)$.

In the following we will show that the theory $\text{SI}^{\omega}(T)$ is a faithful representation of a saturation system extended by the induction rule IND^R and operating on an initial clause set corresponding to a theory T. In other words, we will show that for a sound and refutationally complete saturation system $\mathcal{S}$ and a theory T, the saturation system $\mathcal{S} + \text{IND}^R$ refutes the clause set $CNF(sk^{\exists}(T))$ if and only if $\text{SI}^{\omega}(sk^{\exists}(T))$ is inconsistent. Intuitively, we can see that this is the case because the operation $\text{SI}(T)$ corresponds to a simultaneous application of IND^R to all $L(T)$ formulas. However, by the compactness theorem for first-order logic, only finitely many of these induction formulas actually appear in a proof of the inconsistency of $\text{SI}^{\omega}(sk^{\exists}(T))$. Hence we can derive the same induction axioms with the induction rule IND^R.

Lemma 4.2. Let $\mathcal{S}$ be a sound saturation system and T be a theory. If $\mathcal{S} + \text{IND}^R$ refutes $CNF(sk^{\exists}(T))$, then the theory $\text{SI}^{\omega}(sk^{\exists}(T))$ is inconsistent.

Proof. We show the slightly stronger claim that for a $\mathcal{S} + \text{IND}^R$ deduction $\mathcal{C}_0 \subseteq \mathcal{C}_1 \subseteq \cdots \subseteq \mathcal{C}_j$ from $CNF(sk^{\exists}(T))$, we have $L(\mathcal{C}_j) \subseteq L(\text{SI}^j(sk^{\exists}(T)))$ and $\text{SI}^{\omega}(sk^{\exists}(T)) \models \mathcal{C}_j$. We proceed by induction on j. For the induction base $j = 0$ we have $\text{SI}^{\omega}(sk^{\exists}(T)) \models \mathcal{C}_0$ and $L(\mathcal{C}_0) \subseteq L(\text{SI}^0(sk^{\exists}(T)) = L(sk^{\exists}(T))$, since $\mathcal{C}_0 \subseteq CNF(sk^{\exists}(T))$. For the induction step we consider the clause set $\mathcal{C}_{j+1}$. If $\mathcal{C}_{j+1}$ is obtained by an inference from $\mathcal{S}$, then by the soundness of $\mathcal{S}$ we have $L(\mathcal{C}_{j+1}) = L(\mathcal{C}_j)$ and $\mathcal{C}_j \models \mathcal{C}_{j+1}$. Hence by the induction hypothesis we have $\text{SI}^{\omega}(sk^{\exists}(T)) \models \mathcal{C}_{j+1}$ and clearly $L(\mathcal{C}_{j+1}) = L(\mathcal{C}_j) \subseteq L(\text{SI}^j(sk^{\exists}(T))) \subseteq L(\text{SI}^{j+1}(sk^{\exists}(T)))$. If $\mathcal{C}_{j+1}$ is obtained by an application of the IND^R rule, then $\mathcal{C}_{j+1} = \mathcal{C}_j \cup CNF(sk^{\exists}(I_x\varphi(x, \vec{z})))$, where φ is an $L(\mathcal{C}_j)$ formula. Since $L(\mathcal{C}_j) \subseteq L(\text{SI}^j(sk^{\exists}(T)))$ we have $sk^{\exists}(I_x\varphi(x, \vec{z})) \in \text{SI}^{j+1}(sk^{\exists}(T))$, hence $L(\mathcal{C}_{j+1}) \subseteq \text{SI}^{j+1}(sk^{\exists}(T))$. Moreover since $\text{SI}^j(sk^{\exists}(T)) \models \mathcal{C}_j$ we clearly have $\text{SI}^{j+1}(sk^{\exists}(T)) \models \mathcal{C}_{j+1}$.   □

Lemma 4.3. Let $\mathcal{S}$ be a refutationally complete saturation-based proof system and T be a theory. If the theory $\text{SI}^{\omega}(sk^{\exists}(T))$ is inconsistent, then $\mathcal{S} + \text{IND}^R$ refutes $CNF(sk^{\exists}(T))$.

Proof. Assume that $\mathrm{SI}^\omega(sk^\exists(T))$ is inconsistent, then by the compactness theorem there exists a finite subset S of $\mathrm{SI}^\omega(sk^\exists(T))$ such that S is inconsistent. Furthermore there clearly exist sets $S_0, S_1, \ldots, S_n$ with $n \in \mathbb{N}$ such that $S_0 \subseteq sk^\exists(T)$, $S \subseteq S_n$, and $S_i = S_{i-1} \cup \{sk^\exists(I_i)\}$, with $I_i \in \mathrm{SI}^{i-1}(sk^\exists(T))$-IND and $L(I_i) \subseteq L(S_i)$, for $i = 1, \ldots, n$.

Now we can easily construct a refutation of $CNF(sk^\exists(T))$ in $\mathcal{S} + \mathrm{IND}^R$ by letting $\mathcal{C}_0 = CNF(sk^\exists(T))$, and obtaining $\mathcal{C}_i = \mathcal{C}_{i-1} \cup CNF(sk^\exists(I_i))$ for $i = 1, \ldots, n$ by the IND^R rule. Clearly, $\mathcal{C}_n$ is logically equivalent to S_n, therefore we obtain a refutation from $\mathcal{C}_n$ because of the refutational completeness of $\mathcal{S}$. $\quad\square$

We summarize the results so far in the following proposition.

Proposition 4.4. *Let $\mathcal{S}$ be a sound and refutationally complete saturation-based proof system and T be a theory. Then $\mathcal{S} + \mathrm{IND}^R$ refutes $CNF(sk^\exists(T))$ if and only if the theory $\mathrm{SI}^\omega(sk^\exists(T))$ is inconsistent.*

Proof. An immediate consequence of Lemma 4.2 and Lemma 4.3. $\quad\square$

The theory $\mathrm{SI}^\omega(sk^\exists(T))$ is still not very convenient to work with. By working it a bit we can on the one hand eliminate the recursion that interleaves induction and Skolemization and secondly we can even "factor" out the Skolemization part. We start by analyzing which Skolem symbols occur in the theories generated by $\mathrm{SI}^\omega(\cdot)$. 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.

Lemma 4.5. *Let $\varphi(\vec{y})$ be a formula and u a variable which does not occur in φ. Then $L(sk^\exists(\tilde{I}_u\varphi)) = L(sk^\exists(\varphi \to \varphi))$ and moreover $\vdash sk^\exists(\tilde{I}_u\varphi) \leftrightarrow sk^\exists(\varphi \to \varphi)$.*

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

$$sk^\exists(\tilde{I}_u\varphi) = sk^\forall(\varphi) \wedge sk^\forall(\forall u(\varphi \to \varphi)) \to sk^\exists((\forall u)\varphi)$$
$$= sk^\forall(\varphi) \wedge (sk^\exists(\varphi) \to sk^\forall(\varphi)) \to (\forall u)(sk^\exists(\varphi)).$$

Since $sk^\exists(\varphi \to \varphi) = sk^\forall(\varphi) \to sk^\exists(\varphi)$ we clearly have $L(sk^\exists(\tilde{I}_u\varphi)) = L(sk^\exists(\varphi \to \varphi))$. Furthermore, $sk^\exists(\tilde{I}_u\varphi)$ clearly is logically equivalent to $sk^\exists(\varphi \to \varphi)$. $\quad\square$

The formulas of the form $sk^\exists(\varphi \to \varphi)$ are of interest because they correspond, roughly speaking, to Skolem axioms.

Remark 4.6. The requirement in Lemma 4.5 that the induction formula does not contain the induction variable is peculiar, but convenient to handle. A similar result as Lemma 4.5 can be achieved without this assumption by working, for example, with induction formulas of the form $u = u \wedge \varphi$, where the variable u is not free in the formula φ. In practice a system does usually not intentionally use its induction mechanism to introduce Skolem axioms. Instead some systems (for example [18,19]) provide a lemma rule that introduces the clauses $CNF(sk^\exists(\varphi \to \varphi))$ into the search space.

Lemma 4.7. *Let T be a theory, then $L(\mathrm{SI}^\omega(T)) = sk^\omega(L(T) \cup L_0)$.*

Proof. The inclusion $\subseteq$ is obvious. For the inclusion $\supseteq$ observe that

$$L(\mathrm{SI}^\omega(T)) = \bigcup_{k<\omega} L(\mathrm{SI}^k(T)).$$

Hence, it suffices to show that for every symbol $\sigma \in sk^\omega(L(T) \cup L_0)$, there exists $k \in \mathbb{N}$ such that $\sigma \in L(\mathrm{SI}^{k+1}(T))$. We proceed by induction on the stage of the symbol σ. For the base case let σ have stage 0, then it belongs to $L(T) \cup L_0$ and we already have $\sigma \in L(\mathrm{SI}^1(T))$. Now if $\sigma \in sk^\omega(L(T) \cup L_0)$ has stage $n > 0$, then it is a Skolem symbol of the form $\sigma = \mathfrak{s}_{(Qx)\varphi}$ with $Q \in \{\forall, \exists\}$ and $(Qx)\varphi$ only contains symbols of stage less than n. Hence by the induction hypothesis $L((Qx)\varphi) \subseteq L(\mathrm{SI}^{k+1}(T))$ for some $k \in \mathbb{N}$. Therefore $sk^\exists(I_u(Qx)\varphi) \in \mathrm{SI}^{k+2}(T)$, thus by Lemma 4.5 the symbol $\mathfrak{s}_{(Qx)\varphi}$ belongs to $L(\mathrm{SI}^{k+2}(T))$, where u is a variable that does not occur freely in $(Qx)\varphi$. $\square$

With this in mind we see that $\mathrm{SI}^\omega(T)$ contains the existential Skolemization of the $sk^\omega(L(T))$ induction schema. This allows us to eliminate the iteration of the operator $\mathrm{SI}(\cdot)$ that was used to build up the language of the induction.

Lemma 4.8. *Let T be a theory, then $\mathrm{SI}^\omega(T) \vdash sk^\exists(sk^\omega(L(T) \cup L_0)\text{-}\mathrm{IND})$.*

Proof. Let φ be an $sk^\omega(L(T) \cup L_0)$ formula. By Lemma 4.7 we have $L(\mathrm{SI}^\omega(T)) = \bigcup_{k<\omega} L(\mathrm{SI}^k(T)) = sk^\omega(L(T) \cup L_0)$. Hence, there exists $k \in \mathbb{N}$ such that $L(\varphi) \subseteq L(\mathrm{SI}^k(T))$. Therefore, $\mathrm{SI}^{k+1}(T) \vdash sk^\exists(I_x\varphi)$. $\square$

Again by using Lemma 4.5 it is straightforward to see that by Skolemizing the induction schema $sk^\omega(L)\text{-}\mathrm{IND}$ we actually obtain all the Skolem axioms.

Lemma 4.9. *Let L be a first-order language, then $sk^\exists(sk^\omega(L)\text{-}\mathrm{IND}) \vdash L\text{-}\mathrm{SA}$.*

Proof. Let $\varphi(x, \vec{y})$ be an $sk^\omega(L)$ formula and u be a variable not occurring freely in φ. Work in $sk^\exists(sk^\omega(L)\text{-}\mathrm{IND})$, then in particular we have $sk^\exists(\tilde{I}_u((\forall x)\varphi(x, \vec{y})))$. We apply Lemma 4.5 to $\tilde{I}_u((\forall x)\varphi)$ in order to obtain

$$sk^\forall((\forall x)\varphi(x, \vec{y})) \to sk^\exists((\forall x)\varphi(x, \vec{y})).$$

By Proposition 2.12 we have $\vdash sk^\exists((\forall x)\varphi) \to (\forall x)\varphi$, and hence we obtain

$$sk^\forall((\forall x)\varphi(x, \vec{y})) \to (\forall x)\varphi(x, \vec{y}).$$

Now observe that $sk^\forall((\forall x)\varphi(x, \vec{y})) = sk^\forall(\varphi(\mathfrak{s}_{(\forall x)\varphi(x,\vec{y})}(\vec{y}), \vec{y}))$. Again by Proposition 2.12 we have $\vdash \psi \to sk^\forall(\psi)$ for all $sk^\omega(L)$ formulas ψ. Therefore, we obtain the desired Skolem axiom

$$\varphi(\mathfrak{s}_{(\forall x)\varphi}(\vec{y}), \vec{y})) \to (\forall x)\varphi.$$

Now in order to obtain $\mathrm{SA}_x^\exists \varphi$ we start with $sk^\exists(\tilde{I}_u(\neg(\exists x)\varphi))$ and apply Lemma 4.5 to obtain

$$sk^\forall(\neg(\exists x)\varphi) \to sk^\exists(\neg(\exists x)\varphi).$$

From this we clearly obtain $\neg sk^\exists((\exists x)\varphi) \to \neg sk^\forall((\exists x)\varphi)$, thus, we get

$$sk^\forall((\exists x)\varphi) \to sk^\exists((\exists x)\varphi).$$

Now by Proposition 2.12 we first obtain $(\exists x)\varphi \to sk^\exists(\varphi(\mathfrak{s}_{(\exists x)\varphi}(\vec{y}), \vec{y}))$. Since $sk^\exists((\exists x)\varphi(x, \vec{y})) = sk^\exists(\varphi(\mathfrak{s}_{(\exists x)\varphi}(\vec{y}), \vec{y}))$, we get $(\exists x)\varphi \to \varphi(\mathfrak{s}_{(\exists x)\varphi}(\vec{y}), \vec{y})$ by another application of 2.12. $\square$

Proposition 4.10. *Let T be a theory, then*

$$\mathrm{SI}^\omega(sk^\exists(T)) \equiv (L(T) \cup L_0)\text{-}\mathrm{SA} + T + sk^\omega(L(T) \cup L_0)\text{-}\mathrm{IND}.$$

Proof. First of all observe that $sk^\omega(L(sk^\exists(T)) \cup L_0) = sk^\omega(L(T) \cup L_0)$ and therefore $(L(sk^\exists(T)) \cup L_0)$-SA $= (L(T) \cup L_0)$-SA. For the direction from right to left we observe that

$$(L(T) \cup L_0)\text{-SA} + T + sk^\omega(L(T) \cup L_0)\text{-IND} \vdash$$
$$sk^\exists(T) + sk^\exists(sk^\omega(L(T) \cup L_0)\text{-IND}).$$

With this in mind it is straightforward to see that $(L(T) \cup L_0)$-SA$+T+sk^\omega(L(T) \cup L_0)$-IND $\vdash \mathrm{SI}^\omega(sk^\exists(T))$. For the direction from left to right, we observe that by Lemmas 4.8, 4.9 we have

$$\mathrm{SI}^\omega(sk^\exists(T)) \vdash (L(T) \cup L_0)\text{-SA} + sk^\exists(T) + sk^\exists(sk^\omega(L(T) \cup L_0)\text{-IND}).$$

Hence, by Proposition 2.14 we obtain

$$\mathrm{SI}^\omega(sk^\exists(T)) \vdash (L(T) \cup L_0)\text{-SA} + T + sk^\omega(L(T) \cup L_0)\text{-IND}. \quad \square$$

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

Theorem 4.11. *Let $\mathcal{S}$ be a saturation system, T a theory, and φ an $L(T)$ sentence.*

(i) *If $\mathcal{S}$ is sound and $\mathcal{S} + \mathrm{IND}^R$ refutes $CNF(sk^\exists(T + \neg\varphi))$, then*

$$(L(T) \cup L_0)\text{-SA} + T + sk^\omega(L(T) \cup L_0)\text{-IND} \vdash \varphi.$$

(ii) *If $\mathcal{S}$ is refutationally complete and*

$$(L(T) \cup L_0)\text{-SA} + T + sk^\omega(L(T) \cup L_0)\text{-IND} \vdash \varphi,$$

then $\mathcal{S} + \mathrm{IND}^R$ refutes $CNF(sk^\exists(T + \neg\varphi))$.

Proof. Statement *(i)* is an immediate consequence of Lemma 4.2 and Proposition 4.10 and Statement *(ii)* is an immediate consequence of Lemma 4.3 and Proposition 4.10. $\quad \square$

As a corollary we obtain the soundness of the IND^R rule with respect to the standard model of an arithmetical language.

Corollary 4.12. *Let $\mathcal{S}$ be a sound saturation system, L an arithmetical language, T a sound L theory, and σ an L sentence. If $\mathcal{S} + \mathrm{IND}^R$ refutes the clause set $CNF(sk^\exists(T + \neg\sigma))$, then $\mathbb{N}_L \models \sigma$.*

Proof. By *(i)* of Theorem 4.11 it suffices to show that

$$L\text{-SA} + T + sk^\omega(L)\text{-IND} \sqsubseteq_L \mathrm{Th}(\mathbb{N}_L).$$

This is shown by expanding $\mathbb{N}_L$ by suitable Skolem functions, just as in the traditional model-theoretic proof of Proposition 2.15. The resulting structure satisfies $sk^\omega(L)$-IND since $\mathbb{N}_L$ has induction for all subsets of $\mathbb{N}$. $\quad \square$

We conclude this section with a remark.

Remark 4.13. In the presence of the Skolem axioms every formula is equivalent to an open formula. In particular, for a language L, we have

$$(L \cup L_0)\text{-SA} + \text{Open}(sk^\omega(L))\text{-IND} \vdash sk^\omega(L)\text{-IND}.$$

Thus, we can formulate Theorem 4.11 in a slightly more canonical way, by using $\text{Open}(sk^\omega(L))$-IND in place of $sk^\omega(L)$-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 3.5 into a lemma rule and an induction rule for clause sets.

4.2. Conservativity

In the previous section we have characterized the extension of a sound and refutationally complete saturation system by the induction rule IND^R in terms of a theory with induction over formulas that contain Skolem symbols. This gives rise to the question how the addition of Skolem symbols to the language of the induction schema affects the strength of the system. In particular, can we provide an equivalent Skolem-free induction schema? Let L be a Skolem-free language and T an L theory, then a natural candidate for a Skolem-free characterization of the strength of L-SA $+ T + sk^\omega(L)$-IND is the theory $T + L$-IND.

Question 4.14. Let L be a Skolem-free language and T an L theory, do we have

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

In the following we give a partial answer to the above question. The general case remains open. Our answer relies on the following idea: If a Skolem function is definable in terms of an L formula then wherever the Skolem symbols occur we can instead use its definition to eliminate the symbol.

Definition 4.15. Let L be a Skolem-free language and M an L structure. A function $f : |M|^k \to |M|$ is called L-definable in M if there exists an 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 4.16. Let L be a Skolem-free language. We say that an L structure M has definable Skolem functions if for every L formula $\varphi(\vec{x}, y)$ there exists a function $f : |M|^k \to |M|$ that is L-definable in M and

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

Proposition 4.17. *Let T be a Skolem-free theory. If every model M of $T + L(T)$-IND has definable Skolem functions, then*

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

For the sake of the presentation we have moved the proof of Proposition 4.17 to Appendix A. The proof essentially proceeds by replacing in each model the occurrences of the Skolem symbols by instances of their defining formulas.

In order to illustrate Proposition 4.17 we will consider some practically relevant special cases. An important special case of Proposition 4.17 is when the Skolem functions are definable already in a theory.

Definition 4.18. Let T be a Skolem-free theory. We say that T has definable Skolem functions if for each $L(T)$ formula $\varphi(\vec{x}, y)$, there exists an $L(T)$ formula $\psi(\vec{x}, y)$ such that

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

Proposition 4.19. *Let T be a Skolem-free theory with definable Skolem functions, then every model of T has definable Skolem functions.*

Proof. Let $\varphi(\vec{x}, y)$ be an $L(T)$ formula. Since T has definable Skolem function, there exists $\psi(\vec{x}, y)$ such that $T \vdash (\exists y)\varphi(\vec{x}, y) \rightarrow (\exists! y)(\psi(\vec{x}, y) \wedge \varphi(\vec{x}, y))$. Now let

$$\psi'(\vec{x}, y) := (\neg(\exists y)\varphi(\vec{x}, y) \wedge y = 0) \vee ((\exists y)\varphi(\vec{x}, y) \wedge \psi(\vec{x}, y)).$$

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 $\psi'(\vec{x}, 0)$. 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 $y_1 = y_2 = 0$.   □

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 4.20. Let L be a language, and $\theta(x, y)$ an 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 $(\forall x)(x \prec_\theta y \rightarrow \psi(x, y))$. The total order axioms TO_θ for θ are given by the universal closure of the following formulas

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

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

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

where $\psi(x, \vec{z})$ is an L formula. We define L-$\mathrm{WO}_\theta := \mathrm{TO}_\theta + L$-$\mathrm{LNP}_\theta$.

Proposition 4.21. *Let T be a Skolem-free theory. If there exists an $L(T)$ formula $\theta(x, y)$ such that $T \vdash L(T)$-WO_θ, then T has definable Skolem functions.*

Proof. Let $\varphi(\vec{x}, y)$ be an $L(T)$ 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, for every sound arithmetic theory T containing the symbol $+/2$ with the usual primitive recursive definition of $+$ we have

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

where $\theta := (\exists z)x + z = y$. Therefore, extending the full induction principle by all the Skolem symbols based on such a theory results in a system that proves the same $L(T)$ 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.

5. Restricted induction and Skolemization

In the previous section we have considered some high-level questions about the soundness and conservativity of Skolemization in saturation theorem proving with an unrestricted induction rule. In this section we will focus on the role of Skolem symbols in the more practical setting corresponding to the induction rule Γ-GIND^R given in Definition 3.7, where Γ is a set of formulas. We start by providing in Section 5.1 a representation as a logical theory for sound and refutationally complete saturation systems extended by the induction rule Γ-GIND^R. After that we will make use of that characterization in order to clarify the role of the Skolem symbols in saturation systems extended by the rule Γ-GIND^R mostly under the assumption that Γ is Skolem-free. As already mentioned earlier, the restriction to a Skolem-free Γ deviates from practical systems in which Γ may contain initial Skolem symbols but not induction Skolem symbols. Nevertheless, studying the effect of restricting the occurrences of all Skolem symbols in the induction schema is an interesting theoretical question and allows us to better understand the overall role of Skolem symbols.

5.1. Representation as logical theory

We will now provide a preliminary representation as a logical theory for sound and refutationally complete saturation systems extended by the induction rule Γ-GIND^R. We start by introducing some additional notions that will be used throughout this section.

So far we have considered the traditional induction schema with induction parameters. In the following we introduce a notation for induction without induction parameters. Parameter-free induction schemata have been investigated in mathematical logic [1,32,2,14,27], hence, we adopt a similar notation.

Definition 5.1. Let Γ be a set of formulas, then the parameter-free induction schema for Γ formulas Γ-IND$^-$ is given by Γ-IND$^- \coloneqq \{I_x\gamma \mid \gamma(x) \in \Gamma\}$.

The grounding operator given in the following definition allows us to obtain all instances of a set of formulas obtained by replacing some of the variables by ground terms.

Definition 5.2. Let Γ be a set of formulas and let L be a language. Then we define

$$\Gamma \downarrow L \coloneqq \left\{ \gamma(\vec{x}, t_1, \ldots, t_n) \;\middle|\; \begin{array}{l} \gamma(\vec{x}, z_1, \ldots, z_n) \in \Gamma, \\ t_1, \ldots, t_n \text{ are ground } L \text{ terms} \end{array} \right\}.$$

We can now introduce an operator corresponding to the rule Γ-GIND^R.

Definition 5.3. Let T be a theory and Γ be a set of formulas.

$$\Gamma\text{-GSI}(T) \coloneqq T + sk^\exists((\Gamma \downarrow T)\text{-IND}^-).$$

We define Γ-GSI$^i(T)$ as the i-fold iteration of the Γ-GSI$(\cdot)$ operation. Finally, we define Γ-GSI$^\omega(T) \coloneqq \bigcup_{i<\omega} \Gamma$-GSI$^i(T)$.

It is straightforward to see that $\Gamma\text{-GSI}^{\omega}(\cdot)$ characterizes a sound and refutationally complete saturation-based proof system extended by the induction rule $\Gamma\text{-GIND}^R$.

Proposition 5.4. *Let $\mathcal{S}$ be a sound and refutationally complete saturation-based proof system and T be a theory. Then $\mathcal{S} + \Gamma\text{-GIND}^R$ refutes $CNF(sk^{\exists}(T))$ if and only if $\Gamma\text{-GSI}^{\omega}(sk^{\exists}(T))$ is inconsistent.*

Proof. Analogous to the proof of Proposition 4.4. $\square$

In Section 5.2 we will have a closer look at the role of the Skolem symbols in such saturation systems.

5.2. Induction parameters and Skolem symbols

The induction rule $\Gamma\text{-GIND}^R$ 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, it appears reasonable to begin by comparing sound and refutationally complete saturation systems extended by the rule $\Gamma\text{-GIND}^R$ with the induction schema $\Gamma\text{-IND}^-$. In the setting of linear arithmetic with $\Gamma := \text{Open}(\mathcal{T})$ and $\theta(x,y) := y + x = x \to y = 0$ we readily obtain an example where both systems differ in strength.

Lemma 5.5. *Let $\mathcal{S}$ be a sound and refutationally complete saturation system, then $\mathcal{S} + \text{Open}(\mathcal{T})\text{-GIND}^R$ refutes the clause set $CNF(sk^{\exists}(\mathcal{T} + \neg(\forall x)\theta(x,x)))$.*

Proof. By Proposition 5.4 it suffices to show the inconsistency of the theory

$$\text{Open-GSI}^1(sk^{\exists}(\mathcal{T} + \neg(\forall x)\theta(x,x))).$$

Let $c := \mathfrak{s}_{(\forall x)\theta(x,x)}$, then $\text{Open-GSI}^1(sk^{\exists}(\mathcal{T} + \neg(\forall x)\theta)) \vdash I_x\theta(x,c)$. Hence we now work in the theory $\text{Open-GSI}^1(sk^{\exists}(\mathcal{T} + \neg(\forall x)\theta(x,x)))$ and proceed by induction on x in the formula $\theta(x,c)$. For the base case it suffices to see that $c = c + 0 = 0$ by (A4). For the induction step we assume that $c + x = x \to c = 0$ and $c + s(x) = s(x)$. By (A5) we obtain $s(c + x) = s(x)$ and therefore we obtain $c + x = x$. Hence $c = 0$ by the assumptions. Therefore we now obtain $\theta(c,c)$ and $\neg\theta(c,c)$, that is, $\bot$. $\square$

On the other hand we also have the following.

Lemma 5.6. $\mathcal{T} + \text{Open}(\mathcal{T})\text{-IND}^- \nvdash \theta(x,x)$.

The proof of Lemma 5.6 can be found in Appendix B and consists of the elimination of the symbol p from induction formulas followed by the construction of a model $\mathcal{M}$. The domain of $\mathcal{M}$ consists of elements of the form $(b,i) \in \{0,1\} \times \mathbb{Z}$ such that $b = 0$ implies $i \in \mathbb{N}$. Furthermore, the symbol 0 is interpreted as the element $(0,0)$ and $+$ is interpreted as the operation $(b_1, n_1) +^{\mathcal{M}} (b_2, n_2) = (\max\{b_1, b_2\}, n_1 + n_2)$. Hence, $\mathcal{M} \nvDash \theta((1,0),(1,0))$.

Remark 5.7. We clearly have $\mathcal{T} + \text{Open}(\mathcal{T})\text{-IND} \vdash \theta(x,x)$ by proceeding by induction on x in the formula $\theta(x,y)$. Hence Lemma 5.6 is highly interesting for AITP because it provides us with a simple formula that requires induction on a syntactically more complex formula.

The proof of Lemma 5.5 is reminiscent of the obvious proof of $\theta(x,x)$ in the theory $\mathcal{T} + \text{Open}(\mathcal{T})\text{-IND}$. Thus the proof suggests that the occurrences of Skolem symbols in ground terms of the induction formulas provide some of the strength of induction parameters. In the following we will confirm this intuition (see Theorem 5.22).

We start by showing that the Skolem symbols appearing in the ground terms of the induction axioms of Γ-$\mathrm{GSI}^\omega(sk^\exists(T))$ are not more powerful than induction parameters. This is relatively straightforward because ground terms can be abstracted by induction parameters. In particular, the grounding operation given in Definition 5.2 is absorbed by parameterized induction.

Lemma 5.8. *Let Γ be a set of formulas and L a language, then*

$$\Gamma\text{-IND} \vdash (\Gamma \downarrow L)\text{-IND}.$$

Proof. Observe that $\vdash I_x\varphi(x, \vec{y}, \vec{z}) \to I_x\varphi(x, \vec{y}, \vec{t})$. □

We have announced that this section deals mainly with the case where the set of formulas Γ is Skolem-free. This corresponds to a saturation system that also restricts the occurrences of the initial Skolem symbols. In practical systems this is usually not the case, because the restriction mainly applies to induction Skolem symbols. We briefly address this more general case in the following lemma.

Lemma 5.9. *Let $L \supseteq L_0$ be a first-order language, T an L theory, and Γ a set of L formulas, then*

$$\Gamma\text{-GSI}^\omega(T) \sqsubseteq_L L\text{-SA} + T + \Gamma\text{-IND}.$$

Proof. By the compactness theorem it clearly suffices to show that $\Gamma\text{-GSI}^n(T) \sqsubseteq_L L\text{-SA} + T + \Gamma\text{-IND}$ for all $n \in \mathbb{N}$. We proceed by induction on n and show the slightly stronger claim that $L\text{-SA} + \Gamma\text{-GSI}^n(T) + \Gamma\text{-IND} \sqsubseteq_L L\text{-SA} + T + \Gamma\text{-IND}$, for all $n \in \mathbb{N}$. The base case is trivial since $\Gamma\text{-GSI}^0(T) = T$. For the induction step we have

$$
\begin{aligned}
&L\text{-SA} + \Gamma\text{-GSI}^{n+1}(T) \\
&\qquad =_{\text{Def. } 5.3} L\text{-SA} + \Gamma\text{-GSI}^n(T) + sk^\exists(\Gamma \downarrow \Gamma\text{-GSI}^n(T)\text{-IND}) \\
&\qquad \equiv_{\text{Prop. } 2.14} L\text{-SA} + \Gamma\text{-GSI}^n(T) + \Gamma \downarrow \Gamma\text{-GSI}^n(T)\text{-IND} \\
&\qquad \equiv_{\text{Lem. } 5.8} L\text{-SA} + \Gamma\text{-GSI}^n(T) + \Gamma\text{-IND} \\
&\qquad \sqsubseteq_{\text{ind. hyp.}} L\text{-SA} + T + \Gamma\text{-IND}.\quad □
\end{aligned}
$$

We can now apply the above lemma to the case that is relevant for us in order to show that allowing occurrences of Skolem symbols in ground terms of induction formulas is not stronger than induction parameters.

Proposition 5.10. *Let L be a Skolem-free first-order language, T an L theory, and Γ a set of L formulas, then*

$$\Gamma\text{-GSI}^\omega(sk^\exists(T)) \sqsubseteq_L T + \Gamma\text{-IND}.$$

Proof. Let $L' = L_0 \cup L \cup L(sk^\exists(T))$, then by Lemma 5.9 we have $\Gamma\text{-GSI}^\omega(sk^\exists(T)) \sqsubseteq_{L'} L'\text{-SA} + sk^\exists(T) + \Gamma\text{-IND}$. By Proposition 2.14 we have $L'\text{-SA} + sk^\exists(T) + \Gamma\text{-IND} \equiv L'\text{-SA} + T + \Gamma\text{-IND}$. Since $T + \Gamma\text{-IND}$ is Skolem-free, we have by Proposition 2.15 $L'\text{-SA} + T + \Gamma\text{-IND} \sqsubseteq_L T + \Gamma\text{-IND}$. Hence $\Gamma\text{-GSI}^\omega(sk^\exists(T)) \sqsubseteq_L T + \Gamma\text{-IND}$. □

In particular this shows that $\Gamma\text{-GSI}^\omega(sk^\exists(T))$ is not refutationally stronger than the theory $T + \Gamma\text{-IND}$.

Corollary 5.11. *Let L be a Skolem-free first-order language, T an L theory, and Γ a set of L formulas. If $\Gamma\text{-GSI}^\omega(sk^\exists(T))$ is inconsistent, then $T + \Gamma\text{-IND}$ is inconsistent.*

In the following we will show by a proof-theoretic argument that we even have the converse, that is, ground Skolem terms behave in the refutational setting exactly as induction parameters. Thus, we start by recalling the necessary concepts from proof theory. 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 5.12. Let $\gamma(x, \vec{z})$ be a formula, then we define the sentence $I'_x\gamma$ by

$$I'_x\gamma := (\forall \vec{z})(\exists x)\underbrace{\left((\gamma(0, \vec{z}) \wedge (\gamma(x, \vec{z}) \to \gamma(s(x), \vec{z}))) \to (\forall w)\gamma(w, \vec{z})\right)}_{J_x\gamma(x,\vec{z})}.$$

Let Γ be a set of formulas, then we define $\Gamma\text{-IND}' := \{I'_x\gamma \mid \gamma(x, \vec{z}) \in \Gamma\}$.

This induction schema is clearly equivalent to the usual one given in Definition 2.19.

Lemma 5.13. $\Gamma\text{-IND} \equiv \Gamma\text{-IND}'$.

We will work with the following Gentzen system, which is essentially a variant of the calculus **G1c** given in [42] with atomic logical axioms extended by a cut rule and axioms for equality.

Definition 5.14. A sequent is an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite multisets of formulas.

Definition 5.15. The sequent calculus **G** consists of the following rules
Axioms:

$$\frac{}{A \Rightarrow A}\text{ Ax} \qquad\qquad \frac{}{\bot \Rightarrow}\text{ L}\bot$$

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

Rules for weakening, contraction, and cut:

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

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

$$\frac{\Gamma \Rightarrow \Delta, F \quad F, \Lambda \Rightarrow \Pi}{\Gamma, \Lambda \Rightarrow \Delta, \Pi}\text{ Cut}$$

Rules for logical connectives:

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

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

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

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

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

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 recall some important notions and properties of the calculus $\mathbf{G}$. The calculus $\mathbf{G}$ is sound and complete for first-order logic.

Lemma 5.16. *Let φ be a sentence, then $\vdash \varphi$ if and only if there exists a $\mathbf{G}$ proof of the sequent $\Rightarrow \varphi$.*

Proof. See for example [42]. $\square$

The calculus $\mathbf{G}$ has the following form of cut elimination.

Definition 5.17. In a cut inference the formula F is called the cut formula. We say that a $\mathbf{G}$ proof is in atomic cut-normal form (ACNF, for short) if all of its cut formulas are atomic.

Lemma 5.18. *If a sequent $\Gamma \Rightarrow \Delta$ is provable in $\mathbf{G}$, then it has a $\mathbf{G}$ proof in ACNF.*

Proof. See [42]. $\square$

Definition 5.19. The inference rules $\mathrm{L}\exists$ or $\mathrm{R}\forall$ are called strong quantifier inference rules. Let π be a $\mathbf{G}$ proof, then by $\mathrm{sqi}(\pi)$ we denote the number of strong quantifier inferences in π.

In the argument to follow the number of strong quantifier inferences of a proof will be used as the induction measure.

Lemma 5.20. *Let π be a $\mathbf{G}$ proof in ACNF of the sequent $\Pi, \Sigma \Rightarrow \Delta, \Lambda$, then there exists a proof π' in ACNF of $\Pi, sk^{\exists}(\Sigma) \Rightarrow sk^{\forall}(\Delta), \Lambda$ and $\mathrm{sqi}(\pi') \leq \mathrm{sqi}(\pi)$.*

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. $\square$

Proposition 5.21. *Let T be a theory with $L_0 \subseteq L(T)$ and Γ a set of formulas. If $T + \Gamma\text{-IND}$ is inconsistent, then $\Gamma\text{-GSI}^{\omega}(sk^{\exists}(T))$ is inconsistent.*

Proof. Assume that $T + \Gamma\text{-IND}$ is inconsistent, then clearly $sk^{\exists}(T) + \Gamma\text{-IND}'$ is inconsistent as well. Hence by Lemma 5.16 of $\mathbf{G}$ there exists a proof π in ACNF of a sequent of the form $\Pi, I \Rightarrow$, where Π is a finite

subset of $sk^{\exists}(T)$ and I is a finite subset of Γ-IND$'$. Observe, furthermore, that we can assume without loss of generality that the symbol 0 occurs in Π since $L_0 \subseteq L(T)$.

Let μ be a proof in ACNF of a sequent of the form $\Sigma, I \Rightarrow$ with $\Pi \subseteq \Sigma \subseteq \Gamma$-GSI$^{\omega}(sk^{\exists}(T))$. We proceed by induction on the number of strong quantifier inferences of μ in order to obtain a proof of a sequent $\Sigma' \Rightarrow$ where $\Sigma' \subseteq \Gamma$-GSI$^{\omega}(sk^{\exists}(T))$. If μ does not contain strong quantifier inferences, then we obtain a proof of $\Sigma \Rightarrow$ by permuting inferences on ancestors of I downward. For the induction step assume that μ contains at least one strong quantifier inference. Because μ does not contain non-atomic cuts, we can permute quantifier inferences toward the bottom of the proof without introducing any new strong quantifier inferences. Since Σ is free of strong quantifiers any strong quantifier inference takes place on an ancestor of a formula in I. Hence, by permuting a strong quantifier inference toward the bottom of the proof μ, we obtain a proof ν with sqi$(\nu) \leq$ sqi(μ) of the form

$$
\cfrac{
\cfrac{
\cfrac{
\cfrac{(\nu'(\alpha))}{\Sigma, J_x\varphi(\alpha, \vec{t}), I \Rightarrow}
}{\Sigma, (\exists x) J_x\varphi(x, \vec{t}), I \Rightarrow} \text{ L}\exists
}{\Sigma, (\forall \vec{z})(\exists x) J_x\varphi(x, \vec{z}), I \Rightarrow} \text{ L}\forall^*
}{\Sigma, I \Rightarrow} \text{ LC}
$$

where $\varphi(x, \vec{z})$ is a Γ formula and $\vec{t}$ is a vector of ground terms for which we can assume without loss of generality that $L(\vec{t}) \subseteq L(\Sigma)$. If $\vec{t}$ would contain a symbol σ of I that does not already occur in Σ, then there is a formula $\gamma(\vec{x}) \in \Gamma$ containing σ and we introduce $sk^{\exists}(I_x\gamma(0, \dots, 0))$ into Σ by a left weakening. Now we let

$$
c := \mathfrak{s}_{(\forall x)(\varphi(x, \vec{t}) \to \varphi(s(x), \vec{t}))}.
$$

Then by substituting α by c in ν' we obtain a proof $\mu' = \nu'(c)$ of the sequent $\Sigma, J_x\varphi(c, \vec{t}), I \Rightarrow$. We have sqi$(\mu') =$ sqi$(\nu') =$ sqi$(\nu) - 1$. Then by Lemma 5.20 there exists a proof μ'' in ACNF of $\Sigma, sk^{\exists}(J_x\varphi(c, \vec{t})), I \Rightarrow$ with sqi$(\mu'') \leq$ sqi(μ'). Now observe that $sk^{\exists}(J_x\varphi(c, \vec{t})) = sk^{\exists}(I_x\varphi(x, \vec{t}))$. Finally, we apply the induction hypothesis to μ''' in order to obtain the desired proof.

Now we finish by applying the above procedure to π in order to obtain a proof of $\Sigma \Rightarrow$ with $\Pi \subseteq \Sigma \subseteq \Gamma$-GSI$^{\omega}(sk^{\exists}(T))$. By Lemma 5.16 it follows that Γ-GSI$^{\omega}(sk^{\exists}(T))$ is inconsistent. $\square$

We can summarize the results in the following proposition.

Proposition 5.22. *Let L be a Skolem-free first-order language, T an L theory with $L_0 \subseteq L(T)$, and Γ a set of L formulas, then Γ-GSI$^{\omega}(sk^{\exists}(T))$ is inconsistent if and only if $T + \Gamma$-IND is inconsistent.*

Proof. An immediate consequence of the Propositions 5.10 and 5.21. $\square$

The above result shows 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. We can rephrase the result of Proposition 5.22 as follows.

Theorem 5.23. *Let L be a Skolem-free first-order language, T an L theory, Γ a set of L formulas, φ an L formula such that $L_0 \subseteq L(T) \cup L(\varphi)$, and $\mathcal{S}$ a sound and refutationally complete saturation system. Then $\mathcal{S} + \Gamma$-GIND^R refutes $CNF(sk^{\exists}(T + \neg\varphi))$ if and only if $T + \Gamma$-IND $\vdash \varphi$.*

We have thus obtained a Skolem-free characterization of a sound and refutationally complete saturation-based proof system with the induction rule Γ-GIND^R. We conclude this section with a question about a generalization of Theorem 5.23.

Question 5.24. Consider again the situation of Lemma 5.9, where we have shown that Γ-GSI$^\omega(T)$ is L conservative over L-SA + T + Γ-IND where $L \supseteq L_0$ is a first-order language, T an L theory, and Γ a set of L formulas. This gives rise to the question whether we can characterize a system that allows initial Skolem symbols to occur in the induction formulas without restriction, but restricts the occurrences of induction Skolem symbols in an analogous way to Proposition 5.21. In particular, is Γ-GSI$^\omega(T)$ inconsistent if and only if L-SA + T + Γ-IND is inconsistent?

6. Unprovability

In the previous sections 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 as a logical theory of the induction rule Γ-GIND^R where Γ 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 saturation systems that make use of this induction rule. In Section 6.1 we will provide unprovability results for saturation-based systems that are based on the induction rule Open(L)-GIND^R, where L stands for the language of the initial clause set. Then in Section 6.2 we show that the concrete methods described in [36,23] belong to this family and that therefore we obtain unprovability results for these methods.

6.1. Open induction

The setting of linear arithmetic described in Section 2.3 proves to be a source of very simple and practically relevant unprovability examples. We make use of an elegant characterization proved by Shoenfield [40].

Definition 6.1. By $\mathcal{T}'$ we denote the theory having the axioms of $\mathcal{T}$ together with the axioms

$$x \neq 0 \to x = s(p(x)), \tag{B1}$$

$$x + y = y + x, \tag{B2}$$

$$(x + y) + z = x + (y + z), \tag{B3}$$

$$x + y = x + z \to y = z. \tag{B4}$$

Theorem 6.2 *(Shoenfield [40]).* $\mathcal{T}' \equiv \mathcal{T} + \text{Open}(\mathcal{T})\text{-IND}$.

The following formulas were already studied by Shoenfield in [40]. Their interesting relation to the theory $\mathcal{T}'$ will be crucial for our unprovability results.

Definition 6.3. Let m and n be natural numbers, then we define

$$C_m := (\forall x, y)(m \cdot x = m \cdot y \to x = y)$$

$$D_{m,n} := (\forall x, y)(s^n(m \cdot x) \neq m \cdot y).$$

The proof of [40, Theorem 2] given by Shoenfield can be seen to show that $\mathcal{T} + \text{Open}(\mathcal{T})\text{-IND}$ does not prove any of the formulas C_m and $D_{m,n}$ with $0 < n < m$.

Lemma 6.4 *([40]).* Let $0 < n < m$, then

- $\mathcal{T} + \mathrm{Open}(\mathcal{T})\text{-IND} \nvdash C_m$;
- $\mathcal{T} + \mathrm{Open}(\mathcal{T})\text{-IND} \nvdash D_{m,n}$.

We have now everything at hand to formulate the unprovability result.

Definition 6.5. Let $m, n \in \mathbb{N}$, then the clause sets $\mathcal{X}_m$ and $\mathcal{Y}_{m,n}$ are given by

$$\mathcal{X}_m := CNF(sk^\exists(\mathcal{T}' + \neg C_m)),$$
$$\mathcal{Y}_{m,n} := CNF(sk^\exists(\mathcal{T}' + \neg D_{m,n})).$$

Theorem 6.6. *Let $\mathcal{S}$ be a sound saturation system and $\mathcal{C} \in \{\mathcal{X}_m, \mathcal{Y}_{m,n} \mid 0 < n < m\}$, then $\mathcal{S} + \mathrm{Open}(L(\mathcal{C}))\text{-GIND}^R$ does not refute the clause set $\mathcal{C}$.*

Proof. We consider the case for $\mathcal{C} = \mathcal{X}_m$ with $1 < m$. The other case is treated analogously. Proceed indirectly and assume that $\mathcal{S} + \mathrm{Open}(L(\mathcal{X}_m))\text{-GIND}^R$ refutes $\mathcal{X}_m$. Then by Lemma 5.9 we have

$$L(\mathcal{T})\text{-SA} + sk^\exists(\mathcal{T}') + \mathrm{Open}(L(\mathcal{X}_m))\text{-IND} \vdash sk^\forall(C_m)$$

First of all, observe that $sk^\exists(\mathcal{T}') = \mathcal{T}'$. By applying Proposition 2.14 we obtain

$$L(\mathcal{T})\text{-SA} + sk^\exists(\mathcal{T}') + \mathrm{Open}(L(\mathcal{X}_m))\text{-IND} \vdash C_m.$$

Now observe that since $L(\mathcal{X}_m)$ extends $L(\mathcal{T})$ only by constants, we have $\mathrm{Open}(L(\mathcal{X}_m)) = \mathrm{Open}(L(\mathcal{T})) \downarrow L(\mathcal{X}_m)$ and therefore by Lemma 5.8 we obtain

$$L(\mathcal{T})\text{-SA} + \mathcal{T}' + \mathrm{Open}(L(\mathcal{T}))\text{-IND} \vdash C_m.$$

Hence, $\mathcal{T}'$, $\mathrm{Open}(L(\mathcal{T}))\text{-IND}$ and C_m are Skolem-free, thus, we can apply Proposition 2.15 to obtain

$$\mathcal{T}' + \mathrm{Open}(L(\mathcal{T}))\text{-IND} \vdash C_m.$$

By Theorem 6.2 we furthermore obtain $\mathcal{T} + \mathrm{Open}(L(\mathcal{T}))\text{-IND} \vdash C_m$. This contradicts Lemma 6.4. □

This result begs the question which features a system needs in order to prove the sentences C_m and $D_{m,n}$ for $0 < n < m$. In the following we briefly mention some extensions of the open induction schema that would allow us to overcome our unprovability results. The extensions we suggest are purely 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 [40] that C_m and $D_{m,n}$ with $0 < n < m$ can be proved with parameterized double induction (also known as simultaneous induction) on open formulas.

Definition 6.7. Let $\gamma(x, y, \vec{z})$ be a formula, then the formula $\tilde{I}_{(x,y)}\gamma$ is given by

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

Let Γ be a set of formulas, then the double induction schema $\Gamma\text{-IND}_2$ for Γ formulas is given by $\Gamma\text{-IND}_2 := \{(\forall \vec{z})\tilde{I}_{(x,y)}\gamma \mid \gamma(x, y, \vec{z}) \in \Gamma\}$.

Lemma 6.8. *Let $m, n \in \mathbb{N}$ with $0 < n < m$, then*

(i) $\mathcal{T} + \mathrm{Open}(\mathcal{T})\text{-IND}_2 \vdash C_m;$
(ii) $\mathcal{T} + \mathrm{Open}(\mathcal{T})\text{-IND}_2 \vdash D_{m,n}.$

Proof. Straightforward. □

The second possibility is to extend the induction rule used by the system at least to $\forall_1$ formulas without parameters.

Lemma 6.9. *Let $m, n \in \mathbb{N}$ with $0 < n < m$, then*

(i) $\mathcal{T} + \forall_1(\mathcal{T})\text{-IND}^- \vdash \mathcal{T}';$
(ii) $\mathcal{T} + \forall_1(\mathcal{T})\text{-IND}^- \vdash C_m;$
(iii) $\mathcal{T} + \forall_1(\mathcal{T})\text{-IND}^- \vdash D_{m,n}.$

Proof. The proof of (i) is left as an exercise. For (ii) we work in $\mathcal{T} + \forall_1(\mathcal{T})\text{-IND}^-$ and proceed by induction on the formula $(\forall y)(m \cdot x = m \cdot y \to x = y)$. For the base case we have to show that $m \cdot 0 = m \cdot y \to 0 = y$. By Lemma 2.23 we have $m \cdot 0 = 0$. By (B1) we need to distinguish two cases. If $y = 0$, then we are done, otherwise we obtain a contradiction by (A1). For the induction step we assume $(\forall y)(m \cdot x = m \cdot y \to x = y)$ and $m \cdot s(x) = m \cdot y$. We want to obtain $s(x) = y$. By (A5) and (B2) we obtain $s^m(m \cdot x) = m \cdot s(x) = m \cdot y$. By (B1) we can distinguish two cases. If $y = 0$, then by 2.23 we $s^m(m \cdot x) = 0$, which contradicts (A1). Hence by Lemma 2.22 we have $m \cdot x = m \cdot p(y)$ and it suffices to show $x = p(y)$. By the induction hypothesis we have $m \cdot x = m \cdot p(y) \to x = p(y)$. Thus we obtain $x = p(y)$.

For (iii) we proceed analogously. □

Shoenfield has shown the following interesting theorem.

Theorem 6.10 *([40]).* $\mathcal{T}' + \{C_m \mid 1 < m\} + \{D_{m,n} \mid 0 < n < m\}$ *is complete for quantifier-free formulas.*

From this it follows that at least in the setting of linear arithmetic double induction and parameter-free $\forall_1$ induction are sufficient to prove all true quantifier-free formulas.

In a similar way to what we did in this section we obtain many more unprovability results by using independence results of Shepherdson [39] and Schmerl [37]. However, these results are formulated in the language that besides the symbols of linear arithmetic contains the symbols $\dot{-}/2$ and $\cdot/2$ for the truncated subtraction and multiplication, respectively. The properties that are shown independent of the base theory with open induction express slightly more complicated properties such as the irrationality of the square root of two, Fermat's last theorem for $n = 3$, and similar diophantine equations. Hence, these independence results are currently less practically realistic.

6.2. Literal induction: a case study

In the previous section we have provided unprovability results for sound saturation systems that are extended by the rule $\mathrm{Open}(L)\text{-GIND}^R$, where L is a Skolem-free language. In this section we will show that these results apply to the concrete systems described in [36,23].

In [36] Reger and Voronkov describe an AITP system that extends a sound saturation-based proof system by the induction rule

$$\frac{\{\bar{l}(a) \vee C\} \cup \mathcal{C}}{CNF(\neg sk^{\forall}(l(0) \wedge (\forall x)(l(x) \to l(s(x))))) \vee C} \ \text{Literal-AIND}_1^R$$

where a is a constant, $l(x)$ is a *literal free of a*, and $\bar{l}(a)$ ground. We informally refer to this induction rule as the first analytical literal induction rule. Basically, this induction rule operates as follows: Whenever a clause of the form $\bar{l}(a) \vee C$ is encountered, then the rule generates the clauses corresponding to the induction axiom $I_x l(x)$ and immediately resolves these against $\bar{l}(a) \vee C$. In a practical implementation the rule will not apply to every clause of the form $\bar{l}(a) \vee C$ but only when some additional conditions are satisfied. We call this induction rule analytical because an induction is carried out only for literals that actually are generated during the saturation process. The motivation for choosing the very restricted induction rule Literal-AIND$_1^R$ is to solve problems that require "little" induction reasoning and complex first-order reasoning [36]. 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 [23], however, suggest that this method is unable to deal even with very simple yet practically relevant problems such as

$$x + (x + x) = (x + x) + x.$$

In order to relax the overly restricting analyticity, [23] introduces the following induction rule:

$$\frac{\{\bar{l}(a) \vee C\} \cup \mathcal{C}}{CNF(\neg sk^\forall \left(l(0) \wedge (\forall x)(l(x) \to l(s(x))))\right) \vee C} \text{ Literal-AIND}_2^R$$

where $\bar{l}(x)$ is a literal, a is a constant such that $l(a)$ is ground. This rule reduces the degree of analyticity by allowing induction to 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 challenging formulas such as for example $x + (x + x) = (x + x) + x$ (see [23] for details). It is clear that the rule Literal-AIND$_2^R$ is at least as strong as the rule Literal-AIND$_1^R$. Hence we will in the following concentrate on the rule Literal-AIND$_2^R$.

In the next step we will show how the induction rule Literal-AIND$_2^R$ can be expressed in terms of the restricted induction rule given in Definition 3.7. The proof proceeds in three steps: First we extract the induction axioms that are introduced with Literal-AIND$_2^R$; secondly, we derive these induction axioms with the induction rule of Definition 3.7; finally, we use first-order inferences to reconstruct a refutation.

Lemma 6.11. *Let $\mathcal{S}$ be a sound saturation system and $\mathcal{D}_0, \ldots, \mathcal{D}_n$ and $\mathcal{S} + $ Literal-AIND$_2^R$ deduction. There exist $L(\mathcal{D}_0)$ literals*

$$(l_i(x, c_1, \ldots, c_{i-1}))_{i=1,\ldots,k},$$

where $n < k$ and $c_j = \mathfrak{s}_{(\forall x)(l_j(x,c_1,\ldots,c_{j-1}) \to l_j(s(x),c_1,\ldots,c_{j-1}))}$ for $0 < j \leq k$, such that $L(\mathcal{D}_n) \subseteq L(\mathcal{D}_0) \cup \{c_1, \ldots, c_k\}$ and

$$\mathcal{D}_0 \cup sk^\exists(\{I_x l_i(x, c_1, \ldots, c_{i-1}) \mid 0 < i \leq k\}) \models \mathcal{D}_n.$$

Proof. We start with a straightforward observation. Since the induction rule Literal-AIND$_2^R$ introduces only nullary Skolem symbols, every literal appearing in an $\mathcal{S} + $ Literal-AIND$_2^R$ deduction from $\mathcal{D}_0$ is of the form $l(\vec{x}, \vec{c})$, where $\vec{c}$ is a vector of induction Skolem symbols and $L(l(\vec{x}, \vec{y})) \subseteq L(\mathcal{D}_0)$.

Now let us proceed by induction on the length n of the deduction from $\mathcal{D}_0$. If $n = 0$, then clearly we are done. Now assume that the claim holds up to $\mathcal{D}_n$ and consider $\mathcal{D}_{n+1}$. If $\mathcal{D}_{n+1}$ is derived by from $\mathcal{D}_n$ by an inference from $\mathcal{S}$, then by the soundness of $\mathcal{S}$ we have $L(\mathcal{D}_{n+1}) \subseteq L(\mathcal{D}_n)$ and $\mathcal{D}_n \models \mathcal{D}_{n+1}$. Hence, we are done by applying the induction hypothesis to $\mathcal{D}_n$. If $\mathcal{D}_{n+1}$ is derived from $\mathcal{D}_n$ by Literal-AIND$_2^R$, then $\mathcal{D}_n$ contains a clause $\bar{l}(a, c_1, \ldots, c_k) \vee C$ and $\mathcal{D}_{n+1} = \mathcal{D}_n \cup CNF(sk^\exists(I_x l(x, c_1, \ldots, c_k)))$ and moreover

$L(\mathcal{D}_n) \subseteq L(\mathcal{D}_0) \cup \{1, \ldots, n\}$. We let $l_{k+1}(x, \vec{y}) := l(x, \vec{y})$ and $c_{k+1} = \mathfrak{s}_{(\forall x)(l_k(x,c_1,\ldots,c_k) \to l_k(s(x),c_1,\ldots,c_k))}$. Hence we have $\mathcal{D}_n \cup \{sk^{\exists}(I_x l_{k+1}(x, c_1, \ldots, c_k))\} \models \mathcal{D}_{n+1}$ and $L(\mathcal{D}_{n+1}) = L(\mathcal{D}_n) \cup \{c_{k+1}\}$. Therefore, we have $k + 1 < n + 1$ and $L(\mathcal{D}_{n+1}) \subseteq L(\mathcal{D}_0) \cup \{c_1, \ldots, c_{k+1}\}$ and

$$\mathcal{D}_0 \cup sk^{\exists}\left(\{I_x l_i(x, c_1, \ldots, c_{i-1}) \mid 0 < i \leq k + 1\}\right) \models \mathcal{D}_{n+1}. \quad \square$$

Proposition 6.12. *Let $\mathcal{S}$ be a sound and refutationally complete saturation system and T a theory. If $\mathcal{S} +$ Literal-AIND$_2^R$ refutes $CNF(sk^{\exists}(T))$, then the saturation system $\mathcal{S} +$ Literal$(L(sk^{\exists}(T)))$-GIND^R refutes $CNF(sk^{\exists}(T))$.*

Proof. Assume that $\mathcal{S} +$ Literal-AIND$_2^R$ refutes $CNF(sk^{\exists}(T))$, then by Lemma 6.11 we obtain $k \in \mathbb{N}$ and literals $l_i(x, c_1, \ldots, c_{i-1})$ such that $L(l_i(x, \vec{y})) \subseteq L(sk^{\exists}(T))$ $i = 1, \ldots, k$. Moreover we have

$$CNF(sk^{\exists}(T)) \cup sk^{\exists}(\{I_x l_i(x, c_1, \ldots, c_{i-1}) \mid 0 < i \leq k\}) \models \square.$$

Now we start with the clause set $CNF(sk^{\exists}(T))$ and repeatedly apply the induction rule Literal$(L(sk^{\exists}(T)))$-GIND^R to derive the clause sets

$$\mathcal{D}_m = CNF(sk^{\exists}(T)) \cup CNF(sk^{\exists}(\{I_x l_i(x, c_1, \ldots, c_{i-1}) \mid 0 < i \leq m\})),$$

for $m = 1, \ldots, k$. As shown above, this clause set $\mathcal{D}_k$ is inconsistent, hence, by the refutational completeness of $\mathcal{S}$ we obtain a refutation $\mathcal{E}_0, \ldots, \mathcal{E}_n$ from $\mathcal{D}_k$. Hence the sequence $(\mathcal{D}_1, \ldots, D_k, \mathcal{E}_1, \ldots, \mathcal{E}_n)$ is a $\mathcal{S} +$ Literal$(sk^{\exists}(T))$-GIND^R refutation of $CNF(sk^{\exists}(T))$. $\quad \square$

As an immediate consequence, we can transfer the previously established unprovability results to the concrete method described in [36,23].

Theorem 6.13. *Let $\mathcal{S}$ be a sound and refutationally complete saturation system, then the system $\mathcal{S} +$ Literal-AIND$_2^R$ does neither refute the clause set $\mathcal{X}_m$ nor the clause set $\mathcal{Y}_{m,n}$ for $0 < n < m$.*

Proof. We consider the case for the clause set $\mathcal{X}_m$ with $1 < m$. The other case is analogous. Suppose that $\mathcal{S} +$ Literal-AIND$_2^R$ refutes $\mathcal{X}_m$, then by Proposition 6.12 the saturation system $\mathcal{S} +$ Literal$(L(\mathcal{X}_m))$-GIND^R refutes $\mathcal{X}_m$. This contradicts Theorem 6.6. $\quad \square$

Theorem 6.13 gives us a family of simple and practically relevant clause sets that cannot be proved by the calculi presented in [36,23].

Let us now briefly discuss these results. A possible source of criticism for Theorem 6.13 may be that the underlying independence results (Lemma 6.4) are overly strong. That is they do not exploit the restriction of the induction to literals, but instead rely on the fact that the sentences C_m and $D_{m,n}$ with $0 < n < m$ are already unprovable with induction for all quantifier-free formulas. We can address this point by the following results.

Lemma 6.14. *The theory $\mathcal{T} +$ Literal$(\mathcal{T})$-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 $x = s(p(x))$. $\square$

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

Proposition 6.15. $\mathcal{T} + \mathrm{Literal}(\mathcal{T})\text{-IND} \equiv \mathcal{T} + \mathrm{Open}(\mathcal{T})\text{-IND}$.

Proof. The direction from right to left is obvious. For the direction from left to right follows from Lemma 6.14 and Shoenfield's Theorem (Theorem 6.2). $\square$

The underlying independence results are therefore not too strong and it is not possible to improve the result by taking into account the restriction of the induction to literals. The result may also be interesting from a practical point of view, because induction for literals is much easier to implement efficiently than induction for quantifier-free formulas. It would therefore be interesting to investigate under which conditions induction for quantifier-free formulas collapses to induction for literals. However, we believe that there are practically relevant theories in which the induction schema for literals is strictly weaker than the induction schema for quantifier-free formulas. Such a theory could allow us to provide unprovability results that give a motivation for the development of stronger induction mechanisms.

Another possible source of criticism is that our results focus on abstractions that are quite far from practical reality. Most importantly, we do not exploit the fact that the induction rules $\mathrm{Literal\text{-}AIND}_i^R$ ($i = 1, 2$) attempt induction only for literals of which an instance of the dual literal occurs in the derived clauses. Selecting the induction literals in this way seems to be a strong theoretical and practical restriction. However, this restriction is crucial for current practical systems because it permits an efficient operation of the prover. In practice, the restriction is usually weakened by the usage of heuristics for the selection of induction formulas [23]. Another promising method for discovering induction formulas is introduced in [15,44], but it is unclear how to integrate this efficiently into a saturation-based system. We currently do not have a candidate clause set that exploits the way in which $\mathrm{Literal\text{-}AIND}_i^R$ ($i = 1, 2$) select induction literals, but we plan to investigate this restriction in the future.

On the other hand, working with high-level abstractions allows us to obtain results that are robust against minor refinements of the induction rule from [36] such as the refinement proposed in [23]. Moreover, the underlying independence results together with Lemmas 6.8 and 6.9 suggest natural, yet not necessarily practical, extensions of the induction rule by allowing simultaneous induction on multiple variables or by allowing quantification inside the induction formula.

7. Conclusion, future work, and remarks

In this article we have analyzed a commonly used design principle for the integration of induction into saturation systems that has recently received increased interest [31,29], [18,19], [46], [22], [36,23].

In Section 4, we have considered a general framework for induction over natural numbers in saturation-based provers that extend the language by Skolem symbols. By reducing this induction mechanism to a logical theory (see Theorem 4.11), we have shown that in many relevant cases extending the language of the induction schema by Skolem symbols does not grant any additional power (see Proposition 4.21). Furthermore, we have considered, in Section 5, an induction rule that restricts occurrences of Skolem symbols

to ground terms according to similar restrictions observed in practical systems. We have shown that under this restriction Skolem symbols correspond to induction parameters (see Theorem 5.22). Finally, in Section 6, we have used the results from Section 5 and independence results from the literature on mathematical logic to obtain some practically relevant unprovability results for the systems described in [36,23] (see Theorem 6.13).

We plan to continue the work on induction in saturation-based theorem proving by analyzing the methods developed by Cruanes [18,19], Wand [46] and Echenim and Peltier [22]. We are particularly interested in Cruanes' method because its mode of operation is very similar to the methods described in [36,23]. We suspect that under reasonable assumptions, the induction in Cruanes' system corresponds to the restricted induction rule (see Definition 3.7) over $\forall_1$ formulas. Furthermore, Cruanes' method also allows induction on several formulas simultaneously and introduces definitions by the AVATAR splitting mechanism [45].

Furthermore the work in this article has given rise to a number of questions that we hope to address in the future. In Section 4 we have established some very coarse results concerning the conservativity of extensions of the language of the induction formulas by Skolem symbols. In particular we have shown that in many relevant cases extending the induction schema by Skolem symbols does not result in a more powerful system. We have however left open the general case (see Question 4.14). This question is not proper to induction but is part of a more general question concerning the extension of the language of an axiom schema by Skolem symbols. In Section 5 we have mainly considered the case where the occurrences of all Skolem symbols in the induction formulas are subject to the restriction mentioned above. Practical systems only impose this restriction on Skolem symbols that are generated by the induction rule. We have left open the question about a characterization of these systems (see Question 5.24). Finally, it seems worthwhile to investigate the effects of the analyticity properties of induction rules used in concrete systems such as [36,23] and their interaction with redundancy rules.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

This section provides a proof of Proposition 4.17 and all the related lemmas. The proof essentially proceeds by replacing each occurrence of a Skolem symbol by its definition. We start by showing that we can isolate the occurrences of a given function symbol.

Lemma A.1. *Let $\varphi(\vec{z})$ be a formula and f a function symbol. Then there exists a formula $\psi(\vec{z})$ such that $\vdash \varphi(\vec{z}) \leftrightarrow \psi(\vec{z})$ and f occurs in ψ only in subformulas of the form $x = f(\vec{t})$, where $\vec{t}$ is free of f.*

Proof. We exhaustively apply the equivalence from left to right

$$\vdash \varphi(\vec{t}(u))) \leftrightarrow (\forall x)(x = u \rightarrow \varphi(\vec{t}(x))),$$

where x does not occur freely in $\varphi(\vec{t}(u))$. It is straightforward to see that the logical equivalence of the formula so obtained is preserved and furthermore this transformation terminates because the overall nesting depth of f strictly decreases. □

After isolating a function symbol and assuming that it has a definition we can simply replace all the occurrences by its definition.

Lemma A.2. *Let M be an L structure, $\psi(\vec{x}, y)$ be an L formula such that $M \models (\exists! y)\psi(\vec{x}, y)$. Let furthermore f be a function symbol and $\varphi(\vec{z})$ an $L \cup \{f\}$ formula. Let $M' := M \cup \{f \mapsto f_\psi\}$, 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 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 $M' \models f(\vec{x}) = y \leftrightarrow \psi(\vec{x}, y)$. Now apply Lemma A.1 to φ in order to obtain a formula in which f occurs only as subformulas of the form $y = f(\vec{t})$ with $\vec{t}$ free of f and replace these occurrences with $\psi(\vec{t}, y)$. Clearly the resulting formula is equivalent to φ in M'. □

The assumption that a model has definable Skolem functions only provides us with definitions for Skolem symbols of L formulas. The definitions for other Skolem symbols that are introduced at later stages need to be constructed based on the definitions for symbols of lower stages.

Lemma A.3. *Let L be a Skolem-free first-order language and M an L structure with definable Skolem functions. Then there exists an expansion M' of M to $sk^\omega(L)$ such that $M' \models L$-SA and for each Skolem symbol f of $sk^\omega(L)$ then $f^{M'}$ is 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 $sk^i(L)$ such that for each Skolem symbol f/m of $sk^i(L)$ 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 $f := \mathfrak{s}_{(Qy)\varphi(y, \vec{x})}$ be a Skolem symbol of $sk^{i+1}(L)$, that does not belong to $sk^i(L)$. 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 $sk^i(L)$ for all $j = 1, \ldots, n$. By the induction hypothesis there exist 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 A.2 to the formula φ, there exists an 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 L formula, M has definable Skolem functions, there exists a function $h : |M|^k \to |M|$ and an 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 $f^{M_{i+1}} := h$, 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 $M' := \bigcup_{i \geq 0} M_i$. □

Proving Proposition 4.17 is now just a matter of replacing Skolem symbols in induction formulas by their definitions.

Proof of Proposition 4.17. Let φ be an L formula such that $T + L$-SA $+ sk^\omega(L)$-IND $\vdash \varphi$. We proceed indirectly and assume $T + L$-IND $\nvdash \varphi$. Then there exists a model M of $T + L$-IND such that $M \nvDash \varphi$. By Lemma A.3 there exists an expansion M' of M to $sk^\omega(L)$ such that $M' \models L$-SA and for every Skolem symbol f there exists an 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 $sk^\omega(L)$ formula and consider the induction axiom $I_x\theta(x, \vec{z})$. By Lemma A.2 there exists an 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 $M \models L$-IND, we have $M' \models I_x\theta(x, \vec{z})$. Hence $M' \models sk^\omega(L)$-IND and therefore $M' \models T + L$-SA $+ sk^\omega(L)$-IND but $M' \nvDash \varphi$. Contradiction! □

Appendix B

In this section we provide a model theoretic proof of Lemma 5.6. The difficulty consists in showing that a given structure satisfies the induction schema $\mathrm{Open}(L(\mathcal{T}))$-IND$^-$. In order to address this problem we start by simplifying the language of the induction schema (see Proposition B.6). By L' we denote the language $L(\mathcal{T})$ without p.

Lemma B.1. *The theory $\mathcal{T} + \mathrm{Open}(L(\mathcal{T}))\text{-IND}^-$ proves the following formulas*

$$x = 0 \vee x = s(p(x)). \tag{7}$$

$$x + \overline{k} = \overline{k} + x \tag{8}$$

$$x + \overline{k+1} \neq x, \tag{9}$$

for all $k \in \mathbb{N}$.

Next we show that whenever a p-free term contains a free variable x, then whenever the variable x is substituted for $s(x)$, we can propagate one occurrence of the successor function to the root of the term.

Lemma B.2. *Let $t(x)$ be a non-ground p-free term, then there exists a p-free term $t'(x)$ such that $\mathcal{T} \vdash t(s(x)) = s(t'(x))$.*

Proof. We proceed by induction on the structure of the term t. If $t = x$, then we are done by letting $t' = t$. If $t = s(u(x))$, then u is non-ground and p-free. We let $t' = u(s(x))$, then we have $\mathcal{T} \vdash t(s(x)) = s(u(s(x))) = s(t'(x))$. If $t = u_1 + u_2$, then we have to consider two cases depending on whether u_2 is ground. If u_2 is not ground, then by the induction hypothesis there exists u_2' such that $\mathcal{T} \vdash u_2(s(x)) = s(u_2'(x))$. Then we have $\mathcal{T} \vdash u_1(s(x)) + u_2(s(x)) = u_1(s(x)) + s(u_2'(x)) = s(u_1(s(x)) + u_2'(x))$ and we set $t' = u_1(s(x)) + u_2'$. If u_2 is ground, then u_1 is non-ground and by the induction hypothesis there exists u_1' such that $\mathcal{T} \vdash u_1(s(x)) = s(u_1'(x))$. We have $\mathcal{T} \vdash t(s(x)) = u_1(s(x)) + u_2 = s(u_1'(x)) + \overline{k} = s(s^k(u_1'(x)))$, hence we choose $t' = s^k(u_1')$. $\quad\square$

Now we will show that given a term $t(x)$, we can eliminate the occurrences of p in $t(s^N(x))$ when N is large enough.

Lemma B.3. *Let $t(x)$ be a term, then there exists $N \in \mathbb{N}$ and a p-free term t such that $\mathcal{T} \vdash t(s^N(x)) = t'$.*

Proof. If t is a ground term, then we have $\mathcal{T} \vdash t = \overline{k}$ for some k and we let $t' = \overline{k}$ and $N = 0$. If $t = x$, then we let $N = 0$ and $t = t'$. If $t = s(u)$, where u is a term, then by the induction hypothesis there exists N' and a p-free u' such that $\mathcal{T} \vdash u(s^{N'}(x)) = u'$. Hence we have $\mathcal{T} \vdash t(s^N(x)) = s(u(s^N(x))) = s(u')$. Thus we let $N := N'$ and $t' = s(u')$. If $t = p(u)$, then by the induction hypothesis we have some N' and a p-free u' such that $\mathcal{T} \vdash u(s^{N'}(x)) = u'$. Hence by Lemma B.2 we have $\mathcal{T} \vdash p(u(s^{N'+1}(x))) = p(u'(s(x))) = p(s(u'')) = u''$, for some p-free term u'' and we let $N := N' + 1$ and $t' = u''$. If $t = u_1 + u_2$, then by the induction hypothesis there exists for $i \in \{1, 2\}$ a natural number N_i and a p-free term u_i' such that $\mathcal{T} \vdash u_i(s^{N_i}(x)) = u_i'$. Let $N = \max\{N_1, N_2\}$, then we have $\mathcal{T} \vdash t(s^N(x)) = u_1(s^N(x)) + u_2(s^N(x)) = u_1'(s^{N-N_1}(x)) + u_2'(s^{N-N_2}(x))$, thus we let $t' = u_1'(s^{N-N_1}(x)) + u_2'(s^{N-N_2}(x))$. $\quad\square$

Lemma B.4. *Let $\varphi(x)$ be a formula, then there exists $N \in \mathbb{N}$ and a p-free formula $\varphi'(x)$ such that $\mathcal{T} \vdash \varphi(s^N(x)) \leftrightarrow \varphi'$.*

Proof. Let θ an atom, then θ is of the form $t_1 = t_2$, then apply Lemma B.3 twice in order to obtain N_1, N_2 and the p-free terms $t_1'(x)$ and $t_2'(x)$. Now let $N := \max\{N_1, N_2\}$ and observe that $\mathcal{T} \vdash \theta(s^N(x)) \leftrightarrow t_1'(s^{N-N_1}(x)) = t_2'(s^{N-N_2}(x))$.

Let $\theta_1(x), \ldots, \theta_n(x)$ be all the atoms of φ. Let $i \in \{1, \ldots, n\}$, then apply the argument above to θ_i in order to obtain a natural number M_i and a p-free atom θ_i' such that $\mathcal{T} \vdash \theta(s^{M_i}(x)) \leftrightarrow \theta_i'$. Let $M = \max\{M_i \mid i = 1, \ldots, n\}$ and obtain φ' by replacing in $\varphi(s^M(x))$ every atom $\theta_i(s^M(x))$ by $\theta_i'(s^{M-M_i}(x))$. Clearly we have $\mathcal{T} \vdash \varphi(s^M(x)) \leftrightarrow \varphi'$. $\quad\square$

We can now "factor" the symbols p out of the induction schema. The idea is instead of starting the induction at 0 we start the induction at some $N \in \mathbb{N}$ that is large enough, so that we can eliminate p according to the lemma above.

Lemma B.5. $\mathcal{T} + (\text{B1}) + \text{Open}(L')\text{-IND}^- \vdash \text{Open}(\mathcal{T})\text{-IND}^-$.

Proof. Let $\varphi(x)$ be an $L(\mathcal{T})$ formula. We want to show $I_x\varphi(x)$. By Lemma B.4 above we obtain an $N \in \mathbb{N}$ and a p-free formula ψ such that $\mathcal{T} \vdash \varphi(s^N(x)) \leftrightarrow \psi(x)$. Now we work in $\mathcal{T} + (\text{B1}) + \text{Open}(L')\text{-IND}^-$ and assume $\varphi(0)$ and $\varphi(x) \to \varphi(s(x))$ and we want to show $\varphi(x)$. Hence by a $N - 1$ fold application of Lemma (B1) it suffices to show $\varphi(0), \varphi(\overline{1}), \ldots, \varphi(s^N p^N(x))$. By starting with $\varphi(0)$ and iterating $\varphi(x) \to \varphi(s(x))$ we obtain $\varphi(\overline{n})$ for all $n \in \mathbb{N}$. Hence it remains to show $\varphi(s^N(p^N(x)))$. We proceed by induction on ψ. For the induction base we have to show $\psi(0)$ which is equivalent to $\varphi(\overline{N})$, hence we are done. For the induction step we assume $\psi(x)$ and we have to show $\psi(s(x))$. We have $\psi(x) \leftrightarrow \varphi(s^N(x))$ and by $(\forall x)(\varphi(x) \to \varphi(s(x)))$ we obtain $\varphi(s^N(x)) \to \varphi(s^{N+1}(x))$ thus by modus ponens $\varphi(s^{N+1}(x))$ which is equivalent to $\psi(s(x))$. This completes the induction step. By the induction we thus obtain $\psi(x)$, and in particular $\psi(p^N(x))$ which is equivalent to $\varphi(s^N p^N(x))$. This completes the proof. $\quad\square$

As an immediate consequence of the above lemma we can factor all the occurrences of $p/1$ in the induction formulas into a single axiom.

Proposition B.6. $\mathcal{T} + \text{Open}(L(\mathcal{T}))\text{-IND}^- \equiv \mathcal{T} + (\text{B1}) + \text{Open}(L')\text{-IND}^-$.

Over $\mathbb{Z}$ p-free atoms in one variable represent equations between two linear functions as can be easily seen. Linear functions have the nice property that either they coincide everywhere or else they intersect in at most one point. This property of linear functions allows us to analyze the truth values of a quantifier-free formula in one point. The idea is that this property allows us to define a radius, beyond which an atom behaves on the positive integers just like on the negative integers. We will now define an $L(\mathcal{T})$ structure that has an analogous property.

Definition B.7. Let $\mathcal{M}$ be the $L(\mathcal{T})$ structure whose domain is the set of pairs $(b, n) \in \{0, 1\} \times \mathbb{Z}$ such that $b = 0$ implies $n \in \mathbb{N}$ and that interprets the function symbols 0, s, p, and $+$ as follows

$$0^{\mathcal{M}} = (0, 0),$$

$$s^{\mathcal{M}}((b, n)) = (b, n + 1),$$

$$p^{\mathcal{M}}((0, n)) = (0, n \mathbin{\dot-} 1),$$

$$p^{\mathcal{M}}((1, n)) = (1, n - 1),$$

$$(b_1, n_1) +^{\mathcal{M}} (b_2, n_2) = (\max\{b_1, b_2\}, n_1 + n_2),$$

where $b, b_1, b_2 \in \{0, 1\}$ and $n, n_1, n_2 \in \mathbb{N}$.

It is clear that the structure $\mathcal{M}$ is indeed an $L(\mathcal{T})$ structure.

Lemma B.8. $\mathcal{M} \models \mathcal{T} + (\text{B1})$.

Proof. The element $0^{\mathcal{M}}$ clearly has no predecessor. Furthermore $p^{\mathcal{M}} 0^{\mathcal{M}} = (0, 0) = 0^{\mathcal{M}}$. We have $p^{\mathcal{M}}(s^{\mathcal{M}}(b, n)) = (b, n)$. Moreover $(b, n) +^{\mathcal{M}} (0, 0) = (b, n)$ and

$$(b_1, n_1) +^{\mathcal{M}} s^{\mathcal{M}}((b_2, n_2)) = (b_1, n_1) +^{\mathcal{M}} (b_2, n_2 + 1)$$

$$= (\max\{b_1, b_2\}, n_1 + n_2 + 1) = s^{\mathcal{M}}((\max\{b_1, b_2\}, n_1 + n_2))$$

$$= s^{\mathcal{M}}((b_1, n_1) +^{\mathcal{M}} (b_2, n_2)).$$

Finally, observe that every element that is not $0^{\mathcal{M}}$ has a predecessor. $\quad\square$

Lemma B.9. *Let $t(x)$ be a p-free term containing the variable x, then*

$$t^{\mathcal{M}}((b, n)) = (b, t^{\mathbb{Z}}(n)). \tag{10}$$

Proof. We proceed by induction on the structure of the term t. If t is the variable x, then $t^{\mathcal{M}}(b, n) = (b, n) = (b, t^{\mathbb{Z}}(n))$. If $t = t_1 + t_2$, then either t_1 or t_2 is not ground. If t_1 contains x and t_2 does not contain x, then we have by the induction hypothesis $t_1^{\mathcal{M}}(b, n) = (b, t_1^{\mathbb{Z}}(n))$ and $t_2^{\mathcal{M}}(b, n) = (0, t_2^{\mathbb{N}})$. Hence $t^{\mathcal{M}}(b, n) = (b, t_1^{\mathbb{Z}}(n) + t_2^{\mathbb{N}}) = (b, t^{\mathbb{Z}}(n))$. If both t_1 and t_2 contain x, then we have by the induction hypothesis $t_1^{\mathcal{M}}(b, n) = (b, t_1^{\mathbb{Z}}(n))$ and $t_2^{\mathcal{M}}(b, n) = (b, t_2^{\mathbb{Z}}(n))$. Hence $t^{\mathcal{M}}(b, n) = (b, t_1^{\mathbb{Z}}(n) + t_2^{\mathbb{Z}}(n)) = (b, t^{\mathbb{Z}}(n))$. $\quad\square$

The following lemma expresses the informal idea discussed above that an atom is determined outside of some finite radius.

Lemma B.10. *Let $\theta(x)$ be a p-free atom, then there exists $N \in \mathbb{N}$ such that for all $n \geq N$*

$$\mathcal{M} \models \theta((1, -n)) \Longleftrightarrow \mathbb{N} \models \theta(n). \tag{11}$$

Proof. Let $\theta(x) := (t_1 = t_2)$. If t_1 and t_2 do not contain x, then the claim holds trivially. If t_1 and t_2 both contain x, then by Lemma B.9 we have

$$\mathcal{M} \models \theta((1, -n)) \Leftrightarrow \mathbb{Z} \models \theta(-n), \tag{12}$$

for all $n \in \mathbb{N}$. In $\mathbb{Z}$ the atom θ is an equation between two linear functions. Hence there are two cases to consider. If θ is true in $\mathbb{Z}$ in at most one point, then there exists $N \in \mathbb{N}$ such that for all $m \in \mathbb{Z}$ with $|m| \geq N$ we have $\mathbb{Z} \not\models \theta(m)$. Thus $\mathcal{M} \not\models \theta((1, -m))$ and $\mathbb{N} \not\models \theta(m)$ for $m \geq N$. Otherwise, if θ is true in more than one point of $\mathbb{Z}$, then θ is true everywhere in $\mathbb{Z}$ and we have $\mathcal{M} \models \theta((1, -m))$ and $\mathbb{N} \models \theta(m)$ for all $m \geq 0$. If t_1 contains x, but t_2 does not contain x, then clearly $\mathcal{M} \not\models \theta((1, m))$ for all $m \in \mathbb{Z}$. Moreover $\mathcal{M} \models \theta((0, m))$ if and only if $\mathbb{N} \models \theta(m)$ for all $m \geq 0$. Clearly $\mathbb{N} \models \theta(m)$ for at most one $m \in \mathbb{N}$, hence there exists $N \in \mathbb{N}$ such that $\mathbb{N} \not\models \theta(m)$ for all $m \geq N$. Hence we have $\mathcal{M} \not\models \theta((1, -m))$ and $\mathcal{M} \not\models \theta((0, m))$ for all $m \geq N$. This completes the proof. $\quad\square$

Thanks to the property shown in the lemma above, we can now quite easily show that $\mathcal{M}$ is a model of $\mathrm{Open}(L')\text{-IND}^-$.

Lemma B.11. $\mathcal{M} \models \mathrm{Open}(L')\text{-IND}^-$.

Proof. Let $\varphi(x)$ be a quantifier-free p-free formula and assume that $\mathcal{M} \models \varphi(0)$ and $\mathcal{M} \models \varphi(x) \rightarrow \varphi(s(x))$. Let $(b, n) \in |\mathcal{M}|$. If $b = 0$, then the claim follows by a straightforward induction and the definition of the model $\mathcal{M}$. If $b = 1$, then we consider the atoms of the formula $\varphi(x)$. By applying Lemma B.10 to the atoms of φ we obtain a natural number M such that for all $m \geq M$ we have $\mathcal{M} \models \varphi((1, -m)) \Leftrightarrow \mathcal{M} \models \varphi((0, m))$. Clearly, there exists a natural number n' with $n' \leq n$ and $n' \leq -M$. Then we have $\mathcal{M} \models \varphi((1, n'))$ because we have already shown that $\mathcal{M} \models \varphi((0, -n'))$. By applying repeatedly applying the induction step we then obtain $\mathcal{M} \models \varphi((1, n' + k))$ for all $k \in \mathbb{N}$. In particular we obtain $\mathcal{M} \models \varphi((1, n))$. $\quad\square$

We can now finally give a proof of Lemma 5.6

Proof of Lemma 5.6. By Lemma B.8 and Lemma B.11 we have $\mathcal{M} \models \mathcal{T} + (\text{B1}) + \text{Open}(L')\text{-IND}^-$. Now observe that $(1,0) +^{\mathcal{M}} (1,0) = (1,0)$ but $(1,0) \neq (0,0) = 0^{\mathcal{M}}$. Hence $\mathcal{T} + (\text{B1}) + \text{Open}(L')\text{-IND}^- \nvdash \theta(x,x)$. Hence by Proposition B.6 we obtain $\mathcal{T} + \text{Open}(L(\mathcal{T}))\text{-IND}^- \nvdash \theta(x,x)$. □

References

[1] Zofia Adamowicz, Parameter-free induction, the Matiyasevič theorem and BΣ₁, in: F.R. Drake, J.K. Truss (Eds.), Logic Colloquium '86, in: Studies in Logic and the Foundations of Mathematics, vol. 124, Elsevier, 1987, pp. 1–8.

[2] Lev D. Beklemishev, Parameter free induction and reflection, in: Georg Gottlob, Alexander Leitsch, Daniele Mundici (Eds.), Computational Logic and Proof Theory, in: Lecture Notes in Computer Science, vol. 1289, Springer, 1997, pp. 103–113.

[3] Lev D. Beklemishev, Quantifier-free induction schema and the least element principle, Proc. Steklov Inst. Math. 242 (2003) 50–67.

[4] James Brotherston, Nikos Gorogiannis, Rasmus Lerchedahl Petersen, A generic cyclic theorem prover, in: Ranjit Jhala, Atsushi Igarashi (Eds.), Programming Languages and Systems, in: Lecture Notes in Computer Science, vol. 7705, Springer, 2012, pp. 350–367.

[5] Susanne Biundo, Birgit Hummel, Dieter Hutter, Christoph Walther, The Karlsruhe induction theorem proving system, in: Jörg H. Siekmann (Ed.), 8th International Conference on Automated Deduction, in: Lecture Notes in Computer Science, vol. 230, Springer, 1986, pp. 672–674.

[6] Matthias Baaz, Stefan Hetzl, Daniel Weller, On the complexity of proof deskolemization, J. Symb. Log. 77 (2) (2012) 669–686.

[7] Matthias Baaz, Alexander Leitsch, On Skolemization and proof complexity, Fundam. Inform. 20 (4) (1994) 353–379.

[8] Robert S. Boyer, J. Strother Moore, A Computational Logic, ACM Monograph Series, Academic Press, New York, 1979. Edited by Thomas A. Standish.

[9] James Brotherston, Cyclic proofs for first-order logic with inductive definitions, in: Bernhard Beckert (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods, in: Lecture Notes in Computer Science, vol. 3702, Springer, 2005, pp. 78–92.

[10] Alan Bundy, Andrew Stevens, Frank van Harmelen, Andrew Ireland, Alan Smaill Rippling, A heuristic for guiding inductive proofs, Artif. Intell. 62 (2) (1993) 185–253.

[11] Stefano Berardi, Makoto Tatsuta, Classical system of Martin-Löf's inductive definitions is not equivalent to cyclic proof system, in: Javier Esparza, Andrzej S. Murawski (Eds.), Foundations of Software Science and Computation Structures, in: Lecture Notes in Computer Science, vol. 10203, Springer, 2017, pp. 301–317.

[12] Stefano Berardi, Makoto Tatsuta, Classical system of Martin-Löf's inductive definitions is not equivalent to cyclic proofs, Log. Methods Comput. Sci. 15 (3) (August 2019).

[13] Alan Bundy, Frank van Harmelen, Jane Hesketh, Alan Smaill, Andrew Stevens, A rational reconstruction and extension of recursion analysis, in: N.S. Sridharan (Ed.), Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, MI, USA, August 1989, Morgan Kaufmann, 1989, pp. 359–365.

[14] Andrés Cordón-Franco, Alejandro Fernández-Margarit, Francisco Félix Lara Martín, A note on parameter free Π₁-induction and restricted exponentiation, Math. Log. Q. 57 (5) (2011) 444–455.

[15] Koen Claessen, Moa Johansson, Dan Rosén, Nicholas Smallbone, Automating inductive proofs using theory exploration, in: Maria Paola Bonacina (Ed.), Automated Deduction - CADE-24, in: Lecture Notes in Computer Science, vol. 7898, Springer, 2013, pp. 392–406.

[16] Koen Claessen, Moa Johansson, Dan Rosén, Nicholas Smallbone, TIP: Tons of inductive problems, in: Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, Volker Sorge (Eds.), Intelligent Computer Mathematics, in: Lecture Notes in Computer Science, vol. 9150, Springer, 2015, pp. 333–337.

[17] Hubert Comon, Inductionless induction, in: Alan Robinson, Andrei Voronkov (Eds.), Handbook of Automated Reasoning, vol. 1, North-Holland, Amsterdam, 2001, pp. 913–962, chapter 14.

[18] Simon Cruanes, Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond, PhD thesis École Polytechnique, Palaiseau, France, September 2015.

[19] Simon Cruanes, Superposition with structural induction, in: Clare Dixon, Marcelo Finger (Eds.), Frontiers of Combining Systems, in: Lecture Notes in Computer Science, vol. 10483, Springer, 2017, pp. 172–188.

[20] Gilles Dowek, Skolemization in simple type theory: the logical, the theoretical points of view, in: Christoph Benzmüller, Chad Brown, Jörg Siekmann, Richard Statman (Eds.), Reasoning in Simple Type Theory, in: Studies in Logic, Mathematical Logic and Foundations, 2008. College Publications.

[21] Sebastian Eberhard, Stefan Hetzl, Inductive theorem proving based on tree grammars, Ann. Pure Appl. Log. 166 (6) (2015) 665–700.

[22] Mnacho Echenim, Nicolas Peltier, Combining induction and saturation-based theorem proving, J. Autom. Reason. 64 (2) (2020) 253–294.

[23] Márton Hajdú, Petra Hozzová, Laura Kovács, Johannes Schoisswohl, Andrei Voronkov, Induction with generalization in superposition reasoning, in: Christoph Benzmüller, Bruce R. Miller (Eds.), Intelligent Computer Mathematics, in: Lecture Notes in Computer Science, vol. 12236, Springer, 2020, pp. 123–137.

[24] Stefan Hetzl, Jannik Vierling, Clause set cycles and induction, Log. Methods Comput. Sci. 16 (4) (November 2020) 11.

[25] Stefan Hetzl, Tin Lok Wong, Some observations on the logical foundations of inductive theorem proving, Log. Methods Comput. Sci. 13 (4) (April 2018).

[26] Moa Johansson, Lucas Dixon, Alan Bundy, IsaCoSy: synthesis of inductive theorems, in: Workshop on Automated Mathematical Theory Exploration (Automatheo), 2009.

[27] Emil Jeřábek, Induction rules in bounded arithmetic, Arch. Math. Log. 59 (3) (2020) 461–501.

[28] Moa Johansson, Dan Rosén, Nicholas Smallbone, Koen Claessen Hipster, Integrating theory exploration in a proof assistant, in: Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, Josef Urban (Eds.), Intelligent Computer Mathematics, in: Lecture Notes in Computer Science, vol. 8543, Springer, 2014, pp. 108–122.

[29] Abdelkader Kersani, Preuves par induction dans le calcul de superposition, PhD thesis, Université de Grenoble, October 2014.

[30] Ján Komara, Efficient elimination of Skolem functions in LK^h, Arch. Math. Log. (Nov. 2021).

[31] Abdelkader Kersani, Nicolas Peltier, Combining superposition and induction: a practical realization, in: Pascal Fontaine, Christophe Ringeissen, Renate A. Schmidt (Eds.), Frontiers of Combining Systems, in: Lecture Notes in Computer Science, vol. 8152, Springer, 2013, pp. 7–22.

[32] R. Kaye, J. Paris, C. Dimitracopoulos, On parameter free induction schemas, J. Symb. Log. 53 (4) (1988) 1082–1097.

[33] Miller Dale, A compact representation of proofs, Stud. Log. 46 (4) (1987) 347–370.

[34] Uday S. Reddy, Term rewriting induction, in: Mark E. Stickel (Ed.), 10th International Conference on Automated Deduction, in: Lecture Notes in Computer Science, vol. 449, Springer, 1990, pp. 162–177.

[35] Andrew Reynolds, Viktor Kuncak, Induction for SMT solvers, in: Deepak D'Souza, Akash Lal, Kim Guldstrand Larsen (Eds.), Verification, Model Checking, and Abstract Interpretation, in: Lecture Notes in Computer Science, vol. 8931, Springer, 2015, pp. 80–98.

[36] Giles Reger, Andrei Voronkov, Induction in saturation-based proof search, in: Pascal Fontaine (Ed.), Automated Deduction - CADE 27, in: Lecture Notes in Computer Science, vol. 11716, Springer, 2019, pp. 477–494.

[37] Ulf R. Schmerl, Diophantine equations in fragments of arithmetic, Ann. Pure Appl. Log. 38 (2) (1988) 135–170.

[38] Johannes Schoisswohl, Automated induction by reflection, Master's thesis, Technische Universität Wien, 2020.

[39] John C. Shepherdson, A non-standard model for a free variable fragment of number theory, Bull. Acad. Pol. Sci. XII (2) (1964) 79–86.

[40] Joseph R. Shoenfield, Open sentences and the induction axiom, J. Symb. Log. 23 (1) (1958) 7–12.

[41] Andrew Stevens, A rational reconstruction of Boyer and Moore's technique for constructing induction formulas, in: Yves Kodratoff (Ed.), 8th European Conference on Artificial Intelligence, ECAI'88, Pitmann Publishing, 1988, pp. 565–570.

[42] Anne Sjerp Troelstra, Helmut Schwichtenberg, Basic Proof Theory, second edition, Cambridge Tracts in Theoretical Computer Science, vol. 43, Cambridge University Press, 2000.

[43] Jannik Vierling, Cyclic superposition and induction, Master's thesis, Technische Universität Wien, 2018.

[44] Irene Lobo Valbuena, Moa Johansson, Conditional lemma discovery and recursion induction in Hipster, Electron. Commun. EASST 72 (2015).

[45] Andrei Voronkov, AVATAR: the architecture for first-order theorem provers, in: Armin Biere, Roderick Bloem (Eds.), Computer Aided Verification, in: Lecture Notes in Computer Science, vol. 8559, Springer, 2014, pp. 696–710.

[46] Daniel Wand, Superposition: Types and Induction, PhD thesis, Saarland University, Saarbrücken, Germany, 2017.