Clause Set Cycles and Induction

Stefan Hetzl[1,2] and Jannik Vierling[1,3]

[1]Vienna University of Technology
Institute of Discrete Mathematics and Geometry
[2]stefan.hetzl@tuwien.ac.at
[3]jannik.vierling@tuwien.ac.at

Abstract

In this article we relate a family of methods for automated inductive theorem proving based on cycle detection in saturation-based provers to well-known theories of induction. To this end we introduce the notion of clause set cycles—a formalism abstracting a certain type of cyclic dependency between clause sets. We first show that the formalism of clause set cycles is contained in the theory of $\exists_1$ induction. Secondly, we consider the relation between clause set cycles and the theory of open induction. By providing a finite axiomatization of a theory of triangular numbers with open induction we show that the formalism of clause set cycles is not contained in the theory of open induction. Furthermore, we conjecture that open induction and clause set cycles are incomparable. Finally, we transfer these results to a concrete method of automated inductive theorem proving called the n-clause calculus.

1 Introduction

The subject of automated inductive theorem proving (AITP) aims at automating the process of finding proofs by mathematical induction. AITP is of paramount importance to the formal verification of software and hardware. Every non-trivial program contains loops or recursion, hence its verification requires some inductive reasoning. But also the development of proof assistants can benefit from automated inductive theorem proving by providing hammers that can discharge lemmas automatically.

It is folklore that finding proofs by induction is difficult because of the necessity of non-analyticity of induction formulas. The non-analyticity of

induction formulas can be explained proof-theoretically by the failure of cut-elimination in LK with an induction rule, see [10] for a precise statement. A wide variety of approaches have been proposed to address this problem. Among others there are approaches based on enhancements of saturation-based provers [6, 8, 11], cyclic proofs [2], rippling [4], theory exploration [5], etc. Most of these approaches are rather technical in nature and are thus difficult to analyze formally. Hence, most of the analyses carried out for methods of automated inductive theorem proving are empirical and little is known about the theoretical properties of these methods. We believe that providing formal analyses of these methods will contribute to the theoretical foundations of the subject and thus help in developing better methods.

The work presented in this article continues the analysis of Kersani and Peltier's n-clause calculus started in the second author's master's thesis [14]. In [14] refutations of the n-clause calculus are translated into proofs of the sequent calculus LKID introduced in [3]. This translation makes it possible to read off the induction formulas used by the n-clause calculus. The analysis carried out in [14] operates directly on the n-clause calculus as originally defined in [11] without introducing an intermediary abstraction. Therefore, the results obtained in [14] are complicated by inessential technical details. Furthermore, the technicalities of the n-clause calculus made it difficult to state conjectures clearly. For instance, the n-clause calculus imposes restrictions on the types of function symbols. Because of these restrictions it is already difficult to express simple properties such as for example the associativity of natural numbers. In this article we extend the previous work by introducing an abstraction called clause set cycles. This new formalism abstracts the inessential details of the n-clause calculus, thus, allowing us to carry out a more systematic analysis and to formulate more general conjectures.

The article is structured as follows. In Section 2 we will define the notion of clause set cycle and the associated notion of refutability by a clause set cycle. We will then situate these notions with respect to $\exists_1$ induction, and hence show that the formalism is inherently weak. This result is a generalization of the main theorem (Theorem 6.27) of [14]. In Section 3 we will provide a finite axiomatization for a theory of triangular numbers with open induction. This result will then serve as the main technical result in Section 4, where we will show that the notion of refutability by a clause set cycle is not weaker than open induction. In Section 5 we will show that the n-clause calculus is indeed a special case of the system of clause set cycles and transfer the main results of sections 2 and 4 to the n-clause calculus. We thus answer positively the conjecture of [14] that there exists a clause set that is refutable by the n-clause calculus, but that is not refutable with open induction. As

2

a result we situate the power of the n-clause calculus with respect to the theories of $\exists_1$ induction and open induction.

2 Clause Set Cycles and $\exists_1$ Induction

We work in a many-sorted first-order classical logic. By a many-sorted first-order classical logic we understand a classical first-order logic with a finite set of sorts $\mathcal{S}$. The sorts represent universes and are interpreted as pairwise distinct sets. Each function symbol f has a type of the form $s_1 \times \cdots \times s_n \to s_{n+1}$, where $s_1, \ldots, s_n, s_{n+1} \in \mathcal{S}$ are sorts and $n \geq 0$. Analogously, a predicate symbol P has a type of the form $s_1 \times \cdots \times s_n$, again with $s_1, \ldots, s_n \in \mathcal{S}$ and $n \geq 0$. Let the sort $s \in \mathcal{S}$ be interpreted as the set U_s, then a function symbol f with type as above, will be interpreted as a function that maps elements of $U_{s_1} \times \cdots \times U_{s_n}$ to elements of $U_{s_{n+1}}$. Analogously, a predicate symbol P with type as above is interpreted as a subset of $U_{s_1} \times \cdots \times U_{s_n}$. Each individual variable ranges over a fixed sort. Whenever the sort of a variable is clear from the context, we will not mention it explicitly. Terms and formulas are defined as usual, but function symbols and relation symbols need to be applied to terms that agree with the type of the symbol.

Every language that we consider is supposed to contain at least the sort nat representing the natural numbers, with its function symbols 0 : nat representing the number 0 and s : nat $\to$ nat representing the successor function. In case the language contains only one sort, we specify function symbols by pairs of the form f/n where f is a function symbol and n is a natural number representing the arity of the symbol f. In the following we fix one such language and denote it by L. Formulas, structures, models, truth, validity, $\models$, $\vdash$, etc. are defined as usual.

By η we denote a distinguished variable ranging over the sort nat. We will call this variable η a parameter. The parameter η is mainly used to indicate positions on which arguments by induction take place, that is, the parameter usually plays the role of the induction variable. Usually the parameter η will occur freely, in other words, it will not be bound by quantifiers and therefore behaves similarly to a constant. In the literature a similar concept of parameter is used, with the difference that the parameter is usually a constant (even a Skolem constant). In our case treating the parameter as a variable seems to be more natural, especially when dealing with the language of induction formulas.

Let f be a unary function symbol and t a term. In order to save paren-

theses we write ft for the term $f(t)$. By $f^n t$ we abbreviate the term

$$\underbrace{f(f(\ldots f}_{n \text{ times}}(t)\ldots)).$$

Let $n \in \mathbb{N}$, then by $\bar{n}$ we denote the term $\mathsf{s}^n \mathsf{0}$. Let $t, t_1, \ldots, t_n$ be terms of sort nat and $+$ be a function symbol of type $\mathsf{nat} \times \mathsf{nat} \to \mathsf{nat}$ denoting the addition of natural numbers. For the sake of readability we will use the symbol $+$ as an infix symbol. The expression $t_1 + t_2 + \cdots + t_n$ abbreviates the term $(\ldots (t_1 + t_2) + \ldots) + t_n$ and nt denotes the term

$$\underbrace{t + t + \cdots + t}_{n \text{ times}}.$$

Definition 1 (Literal, Clause, Clause set)**.** *An L formula $l(\vec{x})$ is called an L literal if it is an atom or the negation of an atom. An L formula $C(\vec{x})$ is said to be an L clause if it of the form $\forall \vec{y} \bigvee_{i=1}^{k} l_i$ where $l_i(\vec{x}, \vec{y})$ with $i \in \{1, \ldots, k\}$ is a literal. An L formula $S(\vec{x})$ is called an L clause set if it is of the form $\bigwedge_{i=1}^{m} C_i$ where C_i with $i \in \{1, \ldots, m\}$ is an L clause.*

When the language L is clear from the context we simply say literal, clause, and clause set instead of L literal, L clause, and L clause set.

Let φ and ψ be formulas, then by $\varphi \to \psi$ we abbreviate the formula $\neg \varphi \lor \psi$. In order to save some parentheses we use the standard convention that $\to$ associates to the right and has a lower precedence than $\neg$, $\lor$, and $\land$. For example, the formulas $\neg \varphi \to \psi$, $\varphi \land \theta \to \psi$, $\varphi \to \theta \lor \psi$ are to be read as $(\neg \varphi) \to \psi$, $(\varphi \land \theta) \to \psi$ and $\varphi \to (\theta \lor \psi)$, respectively. For the sake of readability we will often not distinguish between a clause and an equivalent formula up to negation normal form. For example if $a_1, \ldots, a_n$, and $b_1, \ldots, b_m$ are atoms, then we also call the formula

$$(a_1 \land \cdots \land a_n) \to (b_1 \lor \cdots \lor b_m)$$

a clause, because its negation normal form is the clause $\bigvee_{i=1}^{n} \neg a_i \lor \bigvee_{i=1}^{m} b_i$. We usually present a concrete clause set $\bigwedge_{i=1}^{n} C_i$ as the list of clauses C_1, $\ldots$, C_n. Similarly, we will usually present the axioms of a theory as a list of formulas. We are now ready to define the notion of clause set cycles and the related notion of refutability by a clause set cycle.

Definition 2. *An L clause set $S(\eta)$ is called an L clause set cycle if it satisfies the following conditions*

$$S(\mathsf{s}\eta) \models S(\eta), \tag{1}$$

$$S(\mathsf{0}) \models \bot. \tag{2}$$

4

Note that clause set cycles do not operate in some background theory. However, a clause set cycle may contain clauses without free variables and these clauses act as a background theory. A clause set cycle has a natural interpretation as an argument by infinite descent, which we will later explain in terms of induction. Before that, we introduce the notion of refutation by a clause set cycle. A refutation by a clause set cycle consists of a clause set cycle and a case distinction.

Definition 3. *We say that an L clause set $R(\eta)$ is refuted by an L clause set cycle $S(\eta)$ if there exists a natural number n such that*

$$R(\mathsf{s}^n\eta) \models S(\eta), \tag{3}$$

$$R(\overline{k}) \models \bot, \ \textit{for all } k \in \{0, \ldots, n-1\}. \tag{4}$$

If the language is clear from the context we simply speak of clause set cycles and of clause sets refuted by a clause set cycle. Let us consider an example in order to clarify the notions of clause set cycle and refutation by a clause set cycle.

Example 4. *Let P and Q be unary predicate symbols over the sort nat and let $R(\eta)$ be the clause set consisting of the clauses*

$$\mathsf{Q}(\eta),$$
$$\neg\mathsf{Q}(0),$$
$$\forall x\,(\mathsf{Q}(\mathsf{s}x) \rightarrow \mathsf{P}(x)),$$
$$\neg\mathsf{P}(0),$$
$$\forall x\,(\mathsf{P}(\mathsf{s}x) \rightarrow \mathsf{P}(x)).$$

Our goal is to refute the clause set $R(\eta)$ by a clause set cycle.

First we claim that $S(\eta) := \mathsf{P}(\eta) \wedge \neg\mathsf{P}(0) \wedge \forall x\,(\mathsf{P}(\mathsf{s}x) \rightarrow \mathsf{P}(x))$ is a clause set cycle. We start by showing that $S(0) \models \bot$. Observe that $S(0) = P(0) \wedge \neg P(0) \wedge \forall x\,(\mathsf{P}(\mathsf{s}x) \rightarrow \mathsf{P}(x))$. Hence we have $S(0) \models \bot$. It remains to show that $S(\mathsf{s}\eta) \models S(\eta)$. Since $S(\mathsf{s}\eta) = P(\mathsf{s}\eta) \wedge \neg P(0) \wedge \forall x\,(\mathsf{P}(\mathsf{s}x) \rightarrow \mathsf{P}(x))$, it suffices to show that $S(\mathsf{s}\eta) \models \mathsf{P}(\eta)$. We have $S(\mathsf{s}\eta) \models P(\mathsf{s}\eta)$ and $S(\mathsf{s}\eta) \models \forall x\,(\mathsf{P}(\mathsf{s}x) \rightarrow \mathsf{P}(x))$, hence $S(\mathsf{s}\eta) \models \mathsf{P}(\mathsf{s}\eta) \rightarrow \mathsf{P}(\eta)$ and by modus ponens we obtain $S(\mathsf{s}\eta) \models P(\eta)$. Hence $S(\eta)$ is a clause set cycle.

Now we claim that $R(\eta)$ is refuted by the clause set cycle $S(\eta)$. It suffices to show the entailments $R(0) \models \bot$ and $R(\mathsf{s}\eta) \models S(\eta)$. By definition of $R(\eta)$ we have $R(\eta) \models \mathsf{Q}(\eta)$ and $R(\eta) \models \neg\mathsf{Q}(0)$, hence $R(0) \models \mathsf{Q}(0)$ and $R(0) \models \neg\mathsf{Q}(0)$. Therefore $R(0) \models \bot$. For the second entailment we have

$R(\mathsf{s}\eta) \models \mathsf{Q}(\mathsf{s}\eta)$ *and* $R(\mathsf{s}\eta) \models \forall x\,(\mathsf{Q}(\mathsf{s}x) \to \mathsf{P}(x))$, *thus* $R(\mathsf{s}\eta) \models \mathsf{Q}(\mathsf{s}\eta) \to$ $\mathsf{P}(\eta)$. *Hence* $R(\mathsf{s}\eta) \models \mathsf{P}(\eta)$. *We thus have* $R(\mathsf{s}\eta) \models S(\eta)$. *Hence* $R(\eta)$ *is refuted by the clause set cycle* $S(\eta)$ *as claimed.*

The cycles introduced in [11] by Kersani and Peltier are parameterized by two natural numbers $i \geq 0$ and $j \geq 1$, that control the argument by infinite descent. The number i—the offset—is the number at which the argument by infinite descent stops, and j—the step—is the number by which the descent proceeds. Our clause set cycles do not have such parameters and therefore appear somewhat restrictive. Let us now demonstrate how to formulate the concepts of step and offset for clause set cycles.

Definition 5. *Let* $S(\eta)$ *be an L clause set and* $i, j \in \mathbb{N}$ *with* $j \geq 1$. *We say that* $S(\eta)$ *is an L clause set cycle with offset* i *and step* j *if*

$$S(\overline{k + i}) \models \bot, \text{ for } k = 0, \ldots, j-1 \text{ and,}$$
$$S(\mathsf{s}^{i+j}\eta) \models S(\mathsf{s}^i\eta).$$

An L clause set $R(\eta)$ *is refuted by the clause set cycle* $S(\eta)$ *with offset* i *and step* j *if there exists a natural number* n *such that*

$$R(\overline{k}) \models \bot, \text{ for } k = 0, \ldots, n-1,$$
$$R(\mathsf{s}^n\eta) \models S(\mathsf{s}^i\eta).$$

We can now show that clause set cycles simulate clause set cycles with offset and step.

Proposition 6. *Let* $R(\eta)$ *be an L clause set. If* $R(\eta)$ *is refuted by an L clause set cycle* $S(\eta)$ *with offset* i *and step* j, *then* $R(\eta)$ *is refuted by an L clause set cycle.*

Proof. It is straightforward to see that the offset i is inessential, by letting $T(\eta) := S(\mathsf{s}^i\eta)$ because the L clause set $T(\eta)$ clearly is a clause set cycle with offset 0 and step j and refutes $R(\eta)$ with offset 0 and step j. In order to show that an arbitrary step j is inessential as well, we let

$$U(\eta) := \bigvee_{l=0}^{j-1} T(\mathsf{s}^l\eta).$$

We will show that U is a clause set cycle. To show that U satisfies (1), it suffices to observe that by the assumption, we have $T(\overline{k}) \models \bot$ for $k = 0, \ldots, j-1$. Therefore $U(0) \models \bot$. In order to show that U satisfies (2),

we need to consider two cases. First let $l \in \{0, \ldots, j-2\}$, then we have $T(\mathsf{s}^{l+1}\eta) \models T(\mathsf{s}^{l+1}\eta)$, which implies $T(\mathsf{s}^l\mathsf{s}\eta) \models U(\eta)$. Now let $l = j - 1$, then we have $T(\mathsf{s}^j\eta) \models T(\eta)$ by the assumption and, thus, $T(\mathsf{s}^l\mathsf{s}\eta) \models U(\eta)$. Therefore, U is a clause set cycle and R is refuted by U. $\qquad\square$

Clause set cycles thus abstract parameters such as offset and step width and therefore simplify a formal analysis.

Let $\psi(x, \vec{z})$ be a formula where x is a variable of sort nat, then the structural induction axiom $I_x\psi$ is defined by

$$I_x\psi := \forall \vec{z} \, (\psi(0, \vec{z}) \to \forall x \, (\psi(x, \vec{z}) \to \psi(\mathsf{s}x, \vec{z})) \to \forall x \, \psi(x, \vec{z})).$$

By an $\exists_1$ formula we understand a formula of the form $\exists \vec{x} \, \varphi(\vec{x}, \vec{y})$, where φ is quantifier-free. The notion of $\forall_1$ formulas is defined dually to $\exists_1$ formulas. We will now introduce the two theories of induction that are of interest for the study of the formalism of clause set cycles.

Definition 7. *The theories $I\exists_1(L)$ and $I\mathrm{Open}(L)$ are given by*

$$I\exists_1(L) := \{I_x\psi \mid \psi(x, \vec{z}) \text{ is an } \exists_1, \, L \text{ formula with } x : \mathsf{nat}\},$$
$$I\mathrm{Open}(L) := \{I_x\psi \mid \psi(x, \vec{z}) \text{ is a quantifier-free, } L \text{ formula with } x : \mathsf{nat}\}.$$

Whenever the language L is clear from the context or irrelevant, we will write $I\exists_1$, $I\mathrm{Open}$ instead of $I\exists_1(L)$, $I\mathrm{Open}(L)$. Let $\varphi(x)$ be a formula with x of sort nat, then we say that φ is inductive if $\vdash \varphi(0)$ and $\varphi(x) \vdash \varphi(\mathsf{s}x)$.

Let us now consider how the notions of clause set cycles and refutability by a clause set cycle relate to provability in theories of induction. Let $S(x)$ be a clause set cycle, then by (1), (2), and by the completeness of first-order logic, we obtain $\vdash \neg S(0)$ and $\neg S(x) \vdash \neg S(\mathsf{s}x)$. In other words, the formula $\neg S(x)$ is inductive. Since S is a clause set, S is logically equivalent to a $\forall_1$ formula, hence $\neg S(x)$ is logically equivalent to an $\exists_1$ formula. Therefore, we have:

Proposition 8. *Let $S(x)$ be a clause set cycle, then we have $I\exists_1 \vdash \neg S(x)$.*

Intuitively a refutation by a clause set cycle consists of a clause set cycle and a case distinction. The case distinction with $n \in \mathbb{N}$ cases can be formalized as follows:

$$C_n(x) := \left(\bigvee_{i=0}^{n-1} x = \bar{i} \right) \vee \exists y \, x = \mathsf{s}^n(y).$$

Since C_n is clearly inductive and logically equivalent to an $\exists_1$ formula, the formula C_n is provable with $\exists_1$ induction. So we have:

Lemma 9. *Let $n \in \mathbb{N}$, then $I\exists_1 \vdash C_n$.*

Let now $R(x)$ be a clause set refuted by a clause set cycle $S(x)$. Then there exists a natural number $n \in \mathbb{N}$ such that R and S satisfy the conditions (3) and (4). We thus have

$$\vdash \neg R(\bar{i}), \ i = 0, 1, \ldots, n-1, \tag{5}$$

$$\neg S(x) \vdash \neg R(\mathsf{s}^n(x)). \tag{6}$$

By the Lemma above we can proceed in $I\exists_1$ by case distinction on the variable x. If $x = \bar{i}$, then we obtain $\neg R(x)$ by (5). Otherwise if $x = \mathsf{s}^n(x')$ for some x', then by Proposition 8 and (6) we have $\neg R(\mathsf{s}^n(x'))$, thus $\neg R(x)$. We therefore obtain:

Theorem 10. *If a clause set $R(x)$ is refuted by a clause set cycle, then*

$$I\exists_1 \vdash \neg R(x).$$

Refutability by a clause set cycle is thus contained in the theory of $\exists_1$ induction. Therefore, methods for AITP that are based on clause set cycles can not prove statements that require induction on formulas with two or more quantifier alternations. This limitation is due to clause set cycles operating on clause sets instead of some larger set of formulas. Similar limitations may apply to other AITP methods that extend saturation-based provers by induction mechanisms that involve only clauses.

3 Open Induction and Triangular numbers

In this section we will provide a finite, universal axiomatization of a theory of triangular numbers with open induction. This finite axiomatization of the theory of triangular numbers will be used in Section 4, to show that there exists a clause set that is refutable by a clause set cycle but that is not refutable by open induction. The result presented in this section is a generalization of the finite axiomatization for multiplication-free arithmetic with open induction provided by Shoenfield in [13].

Let $n \in \mathbb{N}$, then by $\triangle_n$ we denote the n-th triangular number $\sum_{i=0}^{n} i = n(n+1)/2$. By $L_{\triangleright}$ we denote the one-sorted first-order language consisting of the function symbols $\mathsf{0}/0$, $\mathsf{s}/1$, $\mathsf{p}/1$, $+/2$, and the binary predicate symbol $\triangleright$. The predicate symbol $\triangleright$ will be written in infix notation.

Definition 11. *By $T_\triangleright$ we denote the theory axiomatized by*

$$\forall x \, \mathsf{s}x \neq 0. \tag{A1}$$

$$\mathsf{p}0 = 0. \tag{A2}$$

$$\forall x \, \mathsf{ps}x = x. \tag{A3}$$

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

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

$$0 \,\triangleright\, 0. \tag{A6}$$

$$\forall x \, \forall y \, (x \,\triangleright\, y \rightarrow \mathsf{s}x \,\triangleright\, \mathsf{s}x + y). \tag{A7}$$

$$\forall x \, \forall y \, (\mathsf{s}x \,\triangleright\, \mathsf{s}x + y \rightarrow x \,\triangleright\, y). \tag{A8}$$

$$\forall x \, \forall y \, \forall z \, (x \,\triangleright\, y \wedge x \,\triangleright\, z \rightarrow y = z). \tag{A9}$$

The standard model for this theory is denoted by $\mathbb{N}^\triangleright$. The model $\mathbb{N}^\triangleright$ interprets the symbols $0, \mathsf{s}, \mathsf{p}, +$ in the natural way. The predicate symbol $\triangleright$ is interpreted as the graph of the triangle function i.e. the function associating with each natural number n the triangular number $\triangle_n$.

Lemma 12. *The theory $T_\triangleright + I\mathrm{Open}(L_\triangleright)$ proves the following formulas*

$$\forall x \, (x \neq 0 \rightarrow x = \mathsf{sp}x). \tag{B1}$$

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

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

$$\forall x \, \forall y \, \forall z \, (x + y = x + z \rightarrow y = z). \tag{B4}$$

The axiom (A9) is not redundant for the axiomatization of the theory $T_\triangleright + I\mathrm{Open}(L_\triangleright)$. This can be seen by taking the standard interpretation of the symbols 0, s, p, $+$ and by interpreting $\triangleright$ as the set $\{(n, \triangle_n) \mid n \in \mathbb{N}\} \cup \{(n, \triangle_n + 1) \mid n \in \mathbb{N}\}$.

Definition 13. *By $T_\triangleright^I$ we denote the theory $T_\triangleright + B1 + B2 + B3 + B4$.*

Our axiomatization of the theory $T_\triangleright^I$ is not minimal, because (A3) can be derived from the other axioms. Another simple but important observation is that for every formula $\varphi(y)$ of the language $L_\triangleright$ we have

$$T_\triangleright^I \vdash \varphi(\mathsf{sp}x) \leftrightarrow (x = 0 \wedge \varphi(\mathsf{s}0)) \vee (x \neq 0 \wedge \varphi(x)). \tag{7}$$

We will now show that every formula in $T_\triangleright^I$ is equivalent to some formula that is "simple" in the following sense.

Definition 14. *We call a term or a formula simple if it does not contain the symbol* p.

In the theory $T_{\triangleright}^I$ simple terms have a convenient equivalent representation.

Lemma 15. *Let* $t(x_1, \ldots, x_n)$ *be a simple term, then there exist* $m_1, \ldots, m_n, k \in \mathbb{N}$ *such that* $T_{\triangleright}^I \vdash t = m_1 x_1 + \cdots + m_n x_n + \overline{k}$.

Proof. The term t can be rewritten with (A4), (A5), (B2), and (B3) into the form $m_1 x_1 + \cdots + m_n x_n + \overline{k}$. $\qquad\square$

Whenever we are working in the context of the theory $T_{\triangleright}^I$ we will often and implicitly assume that a simple term is of the form $m_1 x_1 + \cdots + m_n x_n + \overline{k}$.

Proposition 16. *Let* φ *be a formula, then there exists a simple formula* ψ *such that* $T_{\triangleright}^I \vdash \varphi \leftrightarrow \psi$.

Proof. We start with an important observation that we shall repeatedly use throughout the proof. Let t be a term containing the symbol p. Now we work in the theory $T_{\triangleright}^I$. Then by using (A5), (B3), and (B4), it is possible to permute the symbol s in the term $\mathsf{s}(t)$ inwards until it is directly above an occurrence of the symbol p.

The proof consists of three main steps. First, we eliminate occurrences of p in the left-hand arguments of triangle atoms. After that, we eliminate in the resulting formula all occurrences of p in right-hand arguments of triangle atoms without introducing p to left-hand arguments of triangle atoms. Finally, we eliminate the symbol p from equational atoms without introducing p to the triangle atoms.

Let ψ be a formula. We will now show that there exists a formula ψ' such that $T_{\triangleright}^I \vdash \psi \leftrightarrow \psi'$ and moreover ψ' does not contain an occurrence of the symbol p in the left-hand arguments of triangle atoms. We start by defining a measure on formulas that will make the argument more apparent. Let us define $\#_1^n(\psi)$ to be the number of triangle atoms in ψ having exactly n occurrences of p in their left-hand argument. We furthermore let

$$N_1^{\psi} := \max\{n \in \mathbb{N} \mid \#_1^n(\psi) \neq 0\}.$$

Now we can define the measure $\#_1(\psi) := (N_1^{\psi}, \#_1^{N_1^{\psi}}(\psi))$. We proceed by induction on the measure $\#_1(\psi)$ with respect to the natural order on pairs of natural numbers. If $\#_1(\psi) = (0, m)$ for some $m \in \mathbb{N}$, then ψ does not contain the symbol p in the left-hand side argument of its triangle atoms.

Therefore ψ is already the desired formula. Otherwise if $N_1^\psi \neq 0$, then $\#_1(\psi) = (N_1^\psi, \#_1^{N_1^\psi}(\psi))$ with $\#_1^{N_1^\psi}(\psi) \neq 0$. Hence there exists a triangle atom $\varphi := t_1 \vartriangleright t_2$ of ψ having exactly N_1^ψ occurrences of p in its left-hand argument. Let $\varphi' := \mathsf{s}(t_1) \vartriangleright \mathsf{s}(t_1) + t_2$, then by (A7) and (A8) we have $T_\vartriangleright^I \vdash \varphi \leftrightarrow \varphi'$. Obtain a $T_\vartriangleright^I$ equivalent atom φ'' from φ' by permuting in the left-hand argument $\mathsf{s}(t_1)$ the symbol s inwards, as described above, until it is directly above an occurrence of p. Now apply (7) to φ'' in order to obtain a $T_\vartriangleright^I$ equivalent formula φ'''. Note that φ''' has four atoms—two equational atoms and two triangle atoms. The triangle atoms both contain exactly $N_1^\psi - 1$ occurrences of p in their left-hand arguments. Let ψ' be obtained from ψ by replacing φ by φ'''. We then have $T_\vartriangleright^I \vdash \psi' \leftrightarrow \psi$. If $N_1^{\psi'} = N_1^\psi$, then by the above we have $\#_1^{N_1^{\psi'}}(\psi') = \#_1^{N_1^\psi}(\psi') = \#_1^{N_1^\psi}(\psi) - 1$. Hence, $\#_1(\psi') < \#_1(\psi)$. Otherwise we have $N_1^{\psi'} < N_1^\psi$ and therefore $\#_1(\psi') < \#_1(\psi)$. In any case we have $\#_1(\psi') < \#_1(\psi)$, thus we obtain the desired formula by applying the induction hypothesis to ψ'.

By the previous step we can now assume that we work with formulas that do not contain p in the left-hand arguments of triangle atoms. We will now eliminate all the occurrences of p in right hand arguments of triangle atoms. To accomplish this we proceed as in the first step by defining $\#_2^n(\psi)$, N_2^ψ, and $\#_2(\psi)$ in analogy to $\#_1^n(\psi)$, N_1^ψ, and $\#_1(\psi)$ for a formula ψ. For a formula ψ without p in the left-hand arguments of triangle atoms we proceed by induction on $\#_2(\psi)$ as in the first step. If $N_2^\psi = 0$, then ψ does not contain p its triangle atoms. Thus ψ is the desired formula. Otherwise if $N_2^\psi \neq 0$, then $\#_2^{N_2^\psi}(\psi) \neq 0$, that is, there exists an atom $\varphi := t_1 \vartriangleright t_2$ of ψ with exactly N_2^ψ occurrences of p in t_2. We let $\varphi' := \mathsf{s}(t_1) \vartriangleright \mathsf{s}(t_1 + t_2)$. Then $T_\vartriangleright^I \vdash \varphi \leftrightarrow \varphi'$. Since t_1 is assumed to be free of p, the term $\mathsf{s}(t_1 + t_2)$ contains N_2^ψ occurrences of p. Now we proceed in analogy to the first step by moving the outermost s inwards, thus allowing us to eliminate one occurrence of p by making use of the equivalence (7). We therefore obtain a formula ψ' with $T_\vartriangleright^I \vdash \psi \leftrightarrow \psi'$ and $\#_2(\psi') < \#_2(\psi)$.

We can now assume that formulas we work with do not contain p in their triangle atoms. It remains to show that we can eliminate the occurrences of p from equational atoms without introducing triangle atoms containing p. Let ψ be a formula, then we define $\#_3^n(\psi)$ to be the number of equational atoms of ψ containing exactly n occurrences of p. We moreover define N_3^ψ and $\#_3(\psi)$ in analogy to N_1^ψ and $\#_1(\psi)$. The induction base is again trivial. For the induction step, we have $N_3^\psi \neq 0$ and $\#_3^{N_3^\psi}(\psi) \neq 0$. Therefore there

exists an equational atom $\varphi := (t_1 = t_2)$ in ψ with exactly N_3^ψ occurrences of p. We proceed similarly to the previous two steps above, by first replacing an the atom φ by the $T_\triangleright^I$ equivalent atom $\mathsf{s}(t_1) = \mathsf{s}(t_2)$, then moving s inwards, and finally applying the equivalence (7). Let ψ' be the resulting formula. It is easy to see that this does not introduce triangle atoms containing p and moreover we have $\#_3(\psi') < \#_3(\psi)$. We can thus apply the induction hypothesis in order to obtain the desired formula. $\qquad\square$

At this point the reader might wonder why the authors chose to include a function symbol for the predecessor function and go through the technicalities of eliminating the symbol p from formulas instead of providing additional axioms. The idea is to work with purely universal axiomatizations so that one can in particular apply Herbrand's theorem. In order to avoid the symbol p we would have to include an axiom containing an existential quantifier such as $\forall x \, \exists y \, ((x = \mathsf{0} \wedge y = \mathsf{0}) \vee (x \neq \mathsf{0} \wedge x = \mathsf{s}(y)))$. As a consequence, we would later run into similar technicalities when dealing with the this existential quantifier. In fact, the function symbol p is just a Skolem function introduced for the axiom given above.

We have now everything at hand to start with the model theoretic considerations of the theory $T_\triangleright^I$. In the following we fix an arbitrary model $\mathcal{M}$ of the theory $T_\triangleright^I$. Our aim is to show that $\mathcal{M}$ is also a model of open induction over the language $L_\triangleright$. By $\mathbf{0}$, $\mathbf{S}$, $\mathbf{P}$, $+$, and $\blacktriangleright$ we denote the respective interpretations of the symbols $\mathsf{0}$, s, p, $+$, and $\triangleright$ in the model $\mathcal{M}$. We start with a few simple observations about the structure of the model $\mathcal{M}$.

Definition 17. *Let $a, b \in \mathcal{M}$, then we define $a \prec b$ if there exists $n \geq 1$ such that $\mathbf{S}^n a = b$. Accordingly we define $a \preceq b$ if $a \prec b$ or $a = b$. We say that a and b are comparable (in symbols $a \sim b$), if $a \preceq b$ or $b \preceq a$.*

It is not hard to see that the relation $\preceq$ is a partial order and that $\sim$ is an equivalence relation. Let a be an element of $\mathcal{M}$, then by $[a]$ we denote the equivalence class of a under $\sim$. It is easy to see that $\preceq$ is total on $[a]$. Hence classes of comparable elements together with $\preceq$ form chains. Let us now look a bit more closely at these chains. As a simple consequence of (B4) we have $\forall x \, x + \overline{k} \neq x$ for all $k \geq 1$. Therefore the chains of comparable elements are infinite. Consider now the class of elements comparable with $\mathbf{0}$ and let a be an element comparable with $\mathbf{0}$. Then by (A1) we have $a = \mathbf{S}^m \mathbf{0}$ for some natural number m, that is, $\mathbf{0}$ is the least element of this chain. The chain of elements comparable with $\mathbf{0}$ thus looks as follows:

$$\mathbf{0} \prec \mathbf{S}^1 \mathbf{0} \prec \mathbf{S}^2 \mathbf{0} \prec \mathbf{S}^3 \mathbf{0} \prec \dots.$$

This chain is isomorphic to the chain of natural numbers with $\leq$ and the addition. This is why we will call the elements comparable with $\mathbf{0}$, the standard elements (of $\mathcal{M}$). Elements that do not belong to this chain are called non-standard elements (of $\mathcal{M}$).

Before we have a look at the structure of a chains of non-standard elements let us summarize some basic properties of $\mathcal{M}$.

Lemma 18. *Let a, b be elements of $\mathcal{M}$*

1. *If a is a non-standard element, then $\mathbf{P}a$ is a non-standard element.*

2. *If a is a non-standard element and $m \in \mathbb{N}$, then $a = \mathbf{S}^m \mathbf{P}^m a$.*

3. *The element $a + b$ is a standard element if and only if a and b are standard elements.*

Proof. For (1.) let a be a non-standard element of $\mathcal{M}$, then $a \neq \mathbf{0}$. Thus by (B1) the element $\mathbf{P}a$ is also a non-standard element.

For (2.) if a is non-standard, then $a \neq 0$ and by (B1) we have $a = \mathbf{SP}a$. By the (1.) the element $\mathbf{P}a$ is non-standard, so we have $\mathbf{P} = \mathbf{SP}(\mathbf{P}a)$ thus $a = \mathbf{S}^2 \mathbf{P}^2 a$ and so on.

For (3.) consider now an element of the form $a + b$. If a and b are both standard elements, then it is clear that $a + b$ is a standard element. Now suppose that $a + b$ is a standard element and suppose without loss of generality that a is not a standard element. Then there exists $m \in \mathbb{N}$ such that $\mathbf{S}^m \mathbf{0} = a + b = b + a = b + \mathbf{S}^{m+1} \mathbf{P}^{m+1} a$. By (A3) we obtain $\mathbf{0} = \mathbf{SP}^{m+1} a$ which contradicts (A1). Hence a must also be a standard element. $\square$

Let us now consider the chain of elements comparable with a non-standard element a of $\mathcal{M}$. Let n, m be natural numbers with $n < k$, then by Lemma 18 we have $\mathbf{S}^n \mathbf{P}^k a = \mathbf{P}^{k-n} a$. Hence we have $\mathbf{P}^k a \prec \mathbf{P}^{k-n} a$. Let b be comparable with a, then either $a = \mathbf{S}^m b$ or $b = \mathbf{S}^m a$ i.e. $b = \mathbf{P}^m a$ or $b = \mathbf{S}^m a$ for some natural number m. The chain thus has the following structure:

$$\cdots \prec \mathbf{P}^3 a \prec \mathbf{P}^2 a \prec \mathbf{P}^1 a \prec a \prec \mathbf{S}^1 a \prec \mathbf{S}^2 a \prec \mathbf{S}^3 a \prec \ldots.$$

The chain of elements comparable with a is isomorphic to the integers with the order $<$.

We define the language $L_{\triangleright}(\mathcal{M})$ to be the extension of the language $L_{\triangleright}$ by a constant symbol a for every element a of $\mathcal{M}$. We let the $L_{\triangleright}$ structure $\mathcal{M}$ interpret $L_{\triangleright}(\mathcal{M})$ formulas by interpreting for every element a of $\mathcal{M}$ the

constant a as itself. The language $L_\triangleright(\mathcal{M})$ will be especially convenient when we need to insert elements of $\mathcal{M}$ into $L_\triangleright$ terms and $L_\triangleright$ formulas. Let $\varphi(x)$ be an $L_\triangleright(\mathcal{M})$ formula. We call an element a of $\mathcal{M}$ a solution of φ if $\varphi(a)$ is true in $\mathcal{M}$. Similarly we call φ valid in $\mathcal{M}$ if $\varphi(a)$ is true in $\mathcal{M}$ for all elements a of $\mathcal{M}$. In the following we will show the crucial observation that simple atomic formulas are either valid in $\mathcal{M}$ or have only finitely many pairwise comparable solutions.

Proposition 19. *Let $\varphi(x, \vec{y})$ be a simple atomic formula, $\vec{b}$ a vector of elements of $\mathcal{M}$, then either $\varphi(x, \vec{b})$ is valid in $\mathcal{M}$ or $\varphi(x, \vec{b})$ has only finitely many pairwise comparable solutions.*

Proof. Depending on the form of φ we need to distinguish between two cases. If φ is of the form $s = t$, then clearly $\varphi(x, \vec{b})$ is equivalent in $\mathcal{M}$ to $mx + c = nx + d$ for some $c, d \in \mathcal{M}$. The claim then follows from Lemma 1 in [13]. If φ is of the form $s \triangleright t$, then $\varphi(x, \vec{b})$ is equivalent in $\mathcal{M}$ to $mx + c \ \blacktriangleright \ nx + d$ for some $n, m \in \mathbb{N}$ and $c, d \in \mathcal{M}$. We need to consider two cases:

- For $m = 0$, we assume that there are at least two comparable solutions e and $\mathbf{S}^p e$ of $\varphi(x, \vec{b})$ with $p > 0$. We have $c \ \blacktriangleright \ ne + d$ and $c \ \blacktriangleright \ n\mathbf{S}^p e + d$. Therefore by (A9) we have $ne + d = n\mathbf{S}^p e + d$. By (B2) and (A5) we obtain $ne + d = \mathbf{S}^{np}\mathbf{0} + ne + d$. By (B4) we then have $\mathbf{0} = \mathbf{S}^{np}\mathbf{0}$. Hence we clearly have $n = 0$. Thus $ne + d = d$, and therefore $c \ \blacktriangleright \ d$ is true in $\mathcal{M}$. Because of that $c \ \blacktriangleright \ nx + d$ is valid in $\mathcal{M}$.

- For $m > 0$, we will show that there are at most two comparable solutions of $\varphi(x, \vec{b})$. We proceed indirectly and assume that there are at least three pairwise comparable solutions e, $\mathbf{S}^{p_1} e$, and $\mathbf{S}^{p_2} e$ of $\varphi(x, \vec{b})$ with $0 < p_1 < p_2$. Since e is a solution we have $me + c \ \blacktriangleright \ ne + d$. Let $i \in \{1, 2\}$, then iterating (A7) and straightforward rewriting we have

$$\mathbf{S}^{p_i m}(me + c) \ \blacktriangleright \ \mathbf{S}^{\triangle_{p_i m}}\mathbf{0} + p_i m(me + c) + ne + d.$$

Since $\mathbf{S}^{p_i} e$ is a solution of $\varphi(x, \vec{b})$ we have $m(\mathbf{S}^{p_i} e) + c \ \blacktriangleright \ n(\mathbf{S}^{p_i} e) + d$. Therefore by (A5), (A9), (B2) and (B4) we obtain

$$\mathbf{S}^{\triangle_{p_i m}}\mathbf{0} + p_i m(me + c) = \mathbf{S}^{n p_i}\mathbf{0}.$$

Thus the element $me + c$ is a standard element of $\mathcal{M}$. Therefore there exists $k \in \mathbb{N}$ such that $me + c = \mathbf{S}^k \mathbf{0}$. We thus have $\mathbf{S}^{\triangle_{p_i m}}\mathbf{0} + p_i m\mathbf{S}^k \mathbf{0} = \mathbf{S}^{n p_i}\mathbf{0}$. Hence by (A1) and because $\mathbf{S}$ is injective we obtain

$$\triangle_{p_i m} + p_i mk = n p_i.$$

14

Hence $m^2 p_1 + m + 2mk = 2n = m^2 p_2 + m + 2mk$. But since $m \neq 0$, this contradicts the assumption that $p_1 < p_2$. $\qquad\square$

We are now ready to show that $\mathcal{M}$ is a model of open induction over the language $L_\rhd$. The proof is analogous to the proof given in [13]. For the sake of completeness we outline the main steps of the proof.

Theorem 20. *Let $\mathcal{M}$ be a model of $T_\rhd^I$, then $\mathcal{M}$ is a model of $I\mathrm{Open}(L_\rhd)$.*

Proof. Let $\theta(x, \vec{z})$ be a quantifier-free $L_\rhd$ formula. We have to show that $\mathcal{M} \models I_x \theta(x, \vec{z})$. By Proposition 16 we can assume without loss of generality that θ is a simple formula. Let $\vec{b}$ be a vector of elements of $\mathcal{M}$ and let $\theta'(x) := \theta(x, \vec{b})$. Assume that $\theta'(\mathbf{0})$ is true in $\mathcal{M}$ and $\theta'(x) \to \theta'(\mathbf{S}(x))$ is valid in $\mathcal{M}$. Let a be an arbitrary element of $\mathcal{M}$. If a is a standard element, then $a = \mathbf{S}^m \mathbf{0}$ for some natural number m. So by applying the induction step repeatedly, starting with $\theta'(\mathbf{0})$, we obtain $\theta'(a)$.

Now let us consider the case when a is non-standard. Consider the atoms from which $\theta'(x)$ is built. By Proposition 19 there are two types of atoms: those that are valid in $\mathcal{M}$, and those that have at most finitely many (two) pairwise comparable solutions. Valid atoms are true in $\mathcal{M}$ regardless of the choice of x, so we need to consider only the remaining atoms. By successively letting x be a, $\mathbf{P}^1 a$, $\mathbf{P}^2 a$, and so on, we will eventually exhaust all the solutions ($\preceq a$) of any invalid atom of $\theta'(x)$. In other words, by choosing $n \in \mathbb{N}$ large enough, the element $\mathbf{P}^n a$ falsifies all the invalid atoms. The same technique works for standard elements, starting at $\mathbf{0}$ and successively considering $\mathbf{S}^1 \mathbf{0}$, $\mathbf{S}^2 \mathbf{0}$, and so forth. Now by taking $m \in \mathbb{N}$ large enough so that both $\mathbf{P}^m a$ and $\mathbf{S}^m \mathbf{0}$ falsify all the invalid atoms, we observe that $\theta'(\mathbf{S}^m \mathbf{0})$ and $\theta'(\mathbf{P}^m a)$ have the same truth value in $\mathcal{M}$. Therefore, since $\theta'(\mathbf{S}^m \mathbf{0})$ is true in $\mathcal{M}$, also $\theta'(\mathbf{P}^m a)$ is true. Since $\mathbf{S}^m \mathbf{P}^m a = a$, we can simply apply the induction step m times to find that $\theta'(a)$ is true in $\mathcal{M}$. $\quad\square$

The finite, universal axiomatizability of $T_\rhd + I\mathrm{Open}(L_\rhd)$ now follows immediately from Theorem 20, completeness of first-order logic, and from Lemma 12.

Theorem 21. *Let φ be a formula, then $T_\rhd + I\mathrm{Open}(L_\rhd) \vdash \varphi$ if and only if $T_\rhd^I \vdash \varphi$.*

4 Clause Set Cycles and Open Induction

In Section 2 we have shown that refutability by a clause set cycle is contained in the theory of $\exists_1$ induction. The next obvious question to ask is whether

refutability by a clause set cycle is also contained in the theory of open induction. In this section we will provide a negative answer to that question by making use of the finite axiomatizability of the theory of triangular numbers with open induction shown in Section 3. In order to provide such a negative answer it suffices to provide a clause set which is refutable by a clause set cycle, but that is not refutable by open induction. A candidate clause set is readily found.

Definition 22. *We denote by $S_\triangleright(\eta)$ the clause set consisting of the clauses* (A4) – (A7) *and the clause* $\forall y \, \neg \eta \triangleright y$.

Let us denote by $L'_\triangleright$ the language of the clause set $S_\triangleright$. The clause set $S_\triangleright$ expresses that the triangle function is not total.

Lemma 23. *The clause set $S_\triangleright(\eta)$ is refutable by a clause set cycle.*

Proof. By the soundness of first-order logic it suffices to show that

$$S_\triangleright(0) \vdash \bot, \tag{8}$$

and

$$S_\triangleright(\mathsf{s}(\eta)) \vdash S_\triangleright(\eta). \tag{9}$$

For (8) we have $S_\triangleright(0) \vdash 0 \triangleright 0$ and $S_\triangleright(0) \vdash \forall y \, \neg 0 \triangleright y$. Hence $S_\triangleright(0) \vdash \bot$. For (9) we assume $S_\triangleright(\mathsf{s}(\eta))$. The clauses of $S_\triangleright$ not having free variables occur in $S_\triangleright(\mathsf{s}(\eta))$, hence we only need to show that $S_\triangleright(\mathsf{s}(\eta)) \vdash \forall y \, \neg \eta \triangleright y$. Let y be arbitrary, then obtain $S_\triangleright(\mathsf{s}(\eta)) \vdash \neg \mathsf{s}(\eta) \triangleright \mathsf{s}(\eta) + y$. By the contrapositive of (A7) we have $\neg \eta \triangleright y$. Therefore the clause set $S_\triangleright$ is a clause set cycle. Since a clause set cycle is trivially refuted by itself, we are done. $\square$

It now remains to show that $S_\triangleright$ cannot be refuted by open induction. In order to be able to make use of Theorem 21, we will now reformulate the clause set $S_\triangleright$ in terms of a theory of triangular numbers. In the following we denote by $T'_\triangleright$ the theory axiomatized by the formulas (A4) – (A7).

Lemma 24. $I\mathrm{Open}(L'_\triangleright) \vdash \neg S_\triangleright(\eta)$ *if and only if* $T'_\triangleright + I\mathrm{Open}(L'_\triangleright) \vdash \forall x \, \exists y \, x \triangleright y$.

Proof. We have the following chain of equivalences.

$$I\mathrm{Open}(L'_\triangleright) \vdash \neg S_\triangleright(\eta)$$
$$\Leftrightarrow I\mathrm{Open}(L'_\triangleright) \vdash \neg((\mathrm{A4}) \wedge \cdots \wedge (\mathrm{A7}) \wedge \forall y \, \neg \eta \triangleright y)$$
$$\Leftrightarrow I\mathrm{Open}(L'_\triangleright) \vdash \neg((\mathrm{A4}) \wedge \cdots \wedge (\mathrm{A7}) \wedge \neg \exists y \, \eta \triangleright y)$$
$$\Leftrightarrow I\mathrm{Open}(L'_\triangleright) \vdash (\mathrm{A4}) \wedge \cdots \wedge (\mathrm{A7}) \rightarrow \exists y \, \eta \triangleright y$$
$$\Leftrightarrow I\mathrm{Open}(L'_\triangleright) \vdash (\mathrm{A4}) \wedge \cdots \wedge (\mathrm{A7}) \rightarrow \forall x \, \exists y \, x \triangleright y$$

16

By the deduction theorem we thus have $I\mathrm{Open}(L'_\rhd) \vdash \neg S_\rhd(\eta)$ if and only if $T'_\rhd + I\mathrm{Open}(L'_\rhd) \vdash \forall x\, \exists y\, x \rhd y$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

In order to complete the negative answer it clearly suffices to show that $T_\rhd + I\mathrm{Open}(L_\rhd) \nvdash \forall x\, \exists y\, x \rhd y$.

Proposition 25. $T_\rhd + I\mathrm{Open}(L_\rhd) \nvdash \forall x\, \exists y\, x \rhd y$.

Proof. We proceed indirectly and assume that $T_\rhd + I\mathrm{Open}(L_\rhd) \vdash \forall x\, \exists y\, x \rhd y$. By Theorem 21 we then also have $T^I_\rhd \vdash \forall x\, \exists y\, x \rhd y$. Since $T^I_\rhd$ is a universal theory we can apply Herbrand's theorem to obtain terms $t_1(x), \ldots, t_k(x)$ such that

$$T^I_\rhd \vdash \bigvee_{i=1}^{k} x \rhd t_i(x).$$

Clearly $\mathbb{N}^\rhd$ is a model of $T^I_\rhd$ and the triangle function $\rhd^{\mathbb{N}^\rhd}$ is quadratic. Since the terms $t_i(x)$, with $i = 1, \ldots, k$ describe linear functions in $\mathbb{N}^\rhd$, there exists $m \in \mathbb{N}$ such that

$$\mathbb{N}^\rhd \nvDash \bigvee_{i=1}^{k} \overline{m} \rhd t_i(\overline{m}).$$

Contradiction! $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

We would like to point out that it seems possible to obtain the result of Proposition 25 by an alternative argument that relies on an interpretation of the theory $T_\rhd + I\mathrm{Open}(L_\rhd)$ in the subtheory $I\Delta_0$ of Peano arithmetic. The idea is to interpret individuals of $T_\rhd + I\mathrm{Open}(L_\rhd)$ as "unary" numbers, that is, finite sequences of "1"s. In particular 0 would be interpreted as the empty sequence, s as the function that appends a "1" to a finite sequence, $+$ as the function that concatenates two finite sequences, and $\rhd$ as a Δ_0 formula that defines the graph of the "unary" triangle function. Since $I\Delta_0$ codes finite sequences the translation outlined above is indeed an interpretation. By Parikh's theorem the provably total functions of $I\Delta_0$ on "unary" elements have a linear growth rate, whereas the "unary" triangle function has a quadratic growth rate. Hence, the translation of $\forall x\, \exists y\, x \rhd y$ is not provable in $I\Delta_0$ and therefore the formula $\forall x\, \exists y\, x \rhd y$ is not provable in $T_\rhd + I\mathrm{Open}(L_\rhd)$. All the notions necessary to develop the argument outlined above can be found in standard textbooks on first-order arithmetic such as [9].

Corollary 26. *The clause set $S_\rhd(\eta)$ is refutable by a clause set cycle and $I\mathrm{Open}(L'_\rhd) \nvdash \neg S_\rhd(\eta)$.*

To summarize we thus have shown the following theorem.

Theorem 27. *There exists a language L, and an L clause set $S(x)$ refutable by clause set cycles such that $I\text{Open}(L) \not\vdash \neg S(x)$.*

Since refutability by a clause set cycle is not contained in open induction, the next obvious question to ask is whether every clause set that is refutable with open induction is also refutable by a clause set cycle. We believe that this is not the case. Intuitively, this can be explained by the following two points: first clause set cycles do not allow for any free variables and secondly clause set cycles only allow for existential quantification. These two shortcomings of clause set cycles can be demonstrated by the following example. We assume the usual right recursive definition of the addition from which we want to prove the sentence $\varphi \equiv \forall x\, x + (x + x) = (x + x) + x$. To prove φ with open induction, we first prove by open induction on the variable y the inductive formula $\psi(x) \equiv \forall y\, x + (x + y) = (x + x) + y$. The "lemma" ψ can now be used to prove φ by instantiating the universally quantified variable y by x. In this example both ideas mentioned above came into play, that is, the "lemma" ψ contains a free variable and it contains a universal quantifier that is actually used to prove φ.

Conjecture 28. *There exists a language L, and an L clause set $S(x)$ such that $I\text{Open}(L) \vdash \neg S(x)$ but $S(x)$ is not refutable by a clause set cycle.*

5 The n-Clause Calculus: A Case Study

In this section we will use the notion of clause set cycle in order to derive results about a concrete approach for AITP—the n-clause calculus. The n-clause calculus is a formalism for AITP that was introduced by Kersani and Peltier in [11]. This calculus enhances the superposition calculus [1], [12]—a refinement of resolution-paramodulation calculi—by a cycle detection mechanism. This mechanism detects a certain type of cyclic dependencies between the clauses that are derived during the saturation process. Such a cyclic dependency represents an argument by infinite descent and, therefore, represents an unsatisfiable subset of the derived clauses. Once such a cycle is detected the refutation is terminated. The n-clause calculus operates on a syntactically restricted fragment of the logical formalism presented in Section 2. The languages in this section are assumed to contain at least one other sort, say ι, besides the sort nat of natural numbers. Furthermore, the languages should not contain any other function symbols of range nat besides $\mathsf{0}$ and s. By an n-clause we understand a clause of the form $\forall \vec{x}\, (N(\eta, \vec{x}) \lor C(\vec{x}))$,

where $N(\eta, \vec{x})$ is a disjunction of literals of the form $\eta \neq t(\vec{x})$ and C is a disjunction of literals of the form $t \bowtie s$ where $\bowtie \in \{=, \neq\}$, and t, s are terms of sort other than nat. The formula N is called the constraint part of the n-clause. An n-clause set is a conjunction of n-clauses. For the sake of readability we will sometimes identify an n-clause set with the set of its conjuncts. The notion of "cycles" of the n-clause calculus is based on the descent operator $\downarrow_j$ with $j \in \mathbb{N}$.

Definition 29. *Let $i \in \mathbb{N}$, and $\mathcal{C} = \forall \vec{x} \, (N(\eta, \vec{x}) \vee C(\vec{x}))$ an n-clause with $N = \bigvee_{j=1}^{k} \eta \neq t_j$. Then we define $\mathcal{C}\downarrow_i := \forall \vec{x} \, (N(\eta, \vec{x})\downarrow_i \vee C(\vec{x}))$ where $N\downarrow_i := \bigvee_{j=1}^{k} \eta \neq \mathsf{s}^i(t_j)$. For an n-clause set $S = \bigwedge_{j=1}^{m} \mathcal{C}_j$ we define $S\downarrow_i := \bigwedge_{j=1}^{m} \mathcal{C}_j\downarrow_i$.*

Intuitively, the $\downarrow_j$ operation allows us to express that η is replaced by its j-th predecessor. The following lemma states a crucial property of the $\downarrow_j$ operator.

Lemma 30. *Let $S(\eta)$ be clause set and $j \geq 0$, then we have $S\downarrow_j(\mathsf{s}^j\eta) \vdash S(\eta)$.*

The converse of the above entailment does not hold. However it holds in a theory that provides at least the injectivity of the successor function.

Lemma 31. *Let $S(\eta)$ be a clause set and $j \geq 0$, then*

$$\forall x \, \forall y \, (\mathsf{s}x = \mathsf{s}y \rightarrow x = y), S(\eta) \vdash S\downarrow_j(\mathsf{s}^j\eta).$$

We can now introduce the notions of cycle and of refutability by a cycle. For the sake of brevity we consider a simplified variant of the n-clause calculus defined in [11]. Only one of the simplifications imposed by us restricts the power of the formalism. The cycles presented in [11] rely on a decidable entailment relation $\sqsupseteq$ between clauses such that $\mathcal{C} \sqsupseteq \mathcal{D}$ implies $\mathcal{C} \models_{\mathrm{KP}} \mathcal{D}$, where $\models_{\mathrm{KP}}$ is the entailment for the "Kersani-Peltier" standard semantics as presented in [11]. Whenever the original formalism requires $\mathcal{C} \sqsupseteq \mathcal{D}$, we require that $\mathcal{C} \models \mathcal{D}$. On the one hand the relation $\models$ is not decidable, but on the other hand the relation $\models$ allows us to use the completeness of first-order logic. The latter would not be possible with $\sqsupseteq$ because this relation relies on standard semantics for which the completeness theorem does not hold. According to [11] the relation $\sqsupseteq$ is intended to abstract decidable relations such as syntactic equality or subsumption that also satisfy our stronger requirement. Hence our restriction does not rule out any practically relevant instance of the n-clause calculus. Finally, our restriction does not limit the generality of Corollary 36 below, since a similar argument could be used, assuming a suitable choice of $\sqsupseteq$.

19

Definition 32. *Let $R(\eta)$ be an n-clause set. A triple $(i, j, S(\eta))$ with $i, j \in \mathbb{N}$, $j > 0$ and $S \subseteq R$ is a cycle for R if $S \vdash \eta \neq \overline{k}$ for $k = i, \ldots, i + j - 1$ and $S \vdash S\!\downarrow_j$. We say that R is refuted by a cycle if there exists a cycle (i, j, S) for R and $R \vdash \eta \neq \overline{k}$, for $k = 0, \ldots, i - 1$.*

A cycle $(i, j, S(\eta))$ for a clause set $R(\eta)$ is similar to an argument by induction with an offset i and a step with j. Accordingly, the conditions $S \vdash n \neq \overline{k}$ for $k = i, \ldots, i + j - 1$ correspond to the j base cases, whereas the condition $S \vdash S\!\downarrow_j$ corresponds to the step case.

Cycles of the n-clause calculus are thus structurally similar to clause set cycles. As announced in Section 1 we will show that clause set cycles are an abstraction of the cycles of the n-clause calculus. In order to show that every n-clause set refutable by a cycle is also refuted by a clause set cycle it essentially remains to show that the argument by induction with offset i and step j can be turned into an argument by structural induction.

Proposition 33. *Let R be an n-clause set refuted by a cycle, then R is refuted by a clause set cycle.*

Proof. Let $(i, j, S(\eta))$ be a cycle refuting R. Consider the formula

$$T(\eta) := \bigvee_{k=0}^{j-1} S(\mathsf{s}^{k+i}\eta).$$

It is not difficult to see that $T(\eta)$ is logically equivalent to a clause. Since S is a cycle, we have $S(\eta) \vdash \eta \neq \overline{i+k}$ for $k = 0, \ldots, j - 1$. Therefore by instantiating η by $\overline{i+k}$ we obtain $S(\overline{i+k}) \vdash \bot$ for $k = 0, \ldots, j - 1$. Hence we have $T(0) \vdash \bot$.

Let $k \in \{0, \ldots, j - 2\}$, then we clearly have $S(\mathsf{s}^{k+i+1}\eta) \vdash T(\eta)$. Now let $k = j - 1$. Since S is a cycle, we have $S \vdash S\!\downarrow_j$. Thus by Lemma 30 we obtain $S(\mathsf{s}^{i+j}\eta) \vdash S(\mathsf{s}^i\eta)$. Therefore $T(\mathsf{s}\eta) \vdash T(\eta)$. Thus, by the soundness of first-order logic $T(\eta)$ is a clause set cycle. Since $S \subseteq R$ we have $R \vdash S$ and therefore $R(\mathsf{s}^i\eta) \vdash T(\eta)$. Now let $k = 0, \ldots, i - 1$, then since $R \vdash n \neq \overline{k}$ we have $R(\overline{k}) \vdash \bot$. Therefore, by the soundness of first-order logic R is refuted by the clause set cycle T. $\qquad\square$

By the above proposition the notion of refutation of an n-clause set by a cycle is also not stronger than $\exists_1$ induction. In the following we will show that an analogue of Theorem 27 also holds for the n-clause calculus. Let L be the language consisting of the two sorts nat, ι, the function symbols $0 : \mathsf{nat}$, $\mathsf{s} : \mathsf{nat} \to \mathsf{nat}$, $+ : \mathsf{nat} \times \iota \to \iota$, $c : \iota$, and the predicate symbol

$\triangleright : \mathsf{nat} \times \iota$. We will again use the predicate symbol $\triangleright$ as an infix symbol. Let $S(\eta)$ be the L n-clause set consisting of the following n-clauses.

$$0 \; \triangleright \; c, \tag{C1}$$

$$\forall x \, \forall y \, (x \; \triangleright \; y \to \mathsf{s}(x) \; \triangleright \; \mathsf{s}(x) + y), \tag{C2}$$

$$\forall x \, \forall y \, (\eta = x \to \neg x \; \triangleright \; y). \tag{C3}$$

Lemma 34. *The n-clause set $S(\eta)$ is refuted by a cycle.*

Proof. By resolving the clauses (C1) and (C3), we obtain $S \vdash \eta \neq 0$. Resolving the clauses (C2) and (C3) yields $\eta = \mathsf{s}(x) \to \neg x \; \triangleright \; y$. Hence we have $S \vdash S\!\downarrow_1$. Thus the triple $(0, 1, S(\eta))$ is a cycle for $S(\eta)$. Therefore $S(\eta)$ is refuted by a cycle. $\qquad \square$

Let us now investigate whether $S(\eta)$ can be refuted by open induction.

Proposition 35. $I\mathrm{Open}(L) \nvdash \neg S(\eta)$.

Proof. Assume that $I\mathrm{Open}(L) \vdash \neg S(\eta)$. Let L' be the one-sorted language obtained from L by replacing the sort ι by the sort nat. We then have $I\mathrm{Open}(L') \vdash \neg S(\eta)$. By replacing the constant c by 0, we obtain

$$I\mathrm{Open}(L'_\triangleright) \vdash \neg S[c/0](\eta).$$

This implies $I\mathrm{Open}(L'_\triangleright) \vdash \neg S_\triangleright(\eta)$, thus, contradicting Corollary 26. $\qquad \square$

By Lemma 34 and Proposition 35 we thus have:

Corollary 36. *There exists a language L and an L n-clause set $S(\eta)$ refuted by a cycle such that $I\mathrm{Open} \nvdash \neg S(\eta)$.*

6 Conclusion

We have introduced the concept of clause set cycles and the notion of refutability by a clause set cycle. Clause set cycles abstract the analogous concepts of cycle and refutability by a cycle of the n-clause calculus. The main advantage of clause set cycles is their semantic nature, which makes them independent of any inference system. This independence of an inference system allows for a more general analysis of the properties of this type of cycle.

We have explained clause set cycles in terms of theories of induction. We first have shown that refutability by a clause set cycle is contained in the theory of $\exists_1$ induction. On the other hand refutation by a clause set cycle is

not contained in the theory of open induction and we even conjecture that open induction is incomparable with the refutability by a clause set cycle. Finally, we have transferred these results to the n-clause calculus. The results allow us to formally situate the strength of a variant of the n-clause calculus with respect to induction, where we formerly only had empirical evidence. The formal results described in this article improve our understanding of the strength of the approaches for AITP based on clause set cycles and help to direct further research.

As mentioned in the introduction, the analysis of clause set cycles is part of a research program which aims at studying methods for automated inductive theorem proving in order to improve the theoretical foundations of this subject. One of the next questions to consider is how clause set cycles can be extended to handle multiple parameters, how this extension would impact the power of the formalism, and how the addition of parameters can be explained from the perspective of induction. Another question to consider is how the enhancement of superposition by structural induction presented by Cruanes in [6] is related to clause set cycles. A further topic of interest are approaches to AITP based on cyclic sequent calculi such as the calculus introduced by Brotherston and Simpson in [3]. For instance the inductive theorem prover "Cyclist" [2] is based on the cut-free fragment of this cyclic calculus. Recently Das [7] has shown that in the context of arithmetic the logical consequences of cyclic proofs containing only Σ_n formulas are contained in the theory $I\Sigma_{n+1}$. In the setting of arithmetic this result already gives us an upper bound for provers such as "Cyclist", however this bound may be improved by taking into account the cut-freeness of the proofs output by "Cyclist".

Acknowledgments

References

[1] Leo Bachmair and Harald Ganzinger. Rewrite-based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation*, 4(3):217–247, 06 1994.

[2] James Brotherston, Nikos Gorogiannis, and Rasmus L. Petersen. A generic cyclic theorem prover. In Ranjit Jhala and Atsushi Igarashi, editors, *Programming Languages and Systems*, volume 7705 of *Lecture Notes in Computer Science*, pages 350–367. Springer Berlin Heidelberg, 2012.

[3] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 10 2010.

[4] Alan Bundy, Andrew Stevens, Frank van Harmelen, Andrew Ireland, and Alan Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62(2):185 – 253, 1993.

[5] Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. Automating inductive proofs using theory exploration. In Maria Paola Bonacina, editor, *Automated Deduction – CADE-24*, volume 7898 of *Lecture Notes in Computer Science*, pages 392–406. Springer Berlin Heidelberg, 2013.

[6] Simon Cruanes. Superposition with structural induction. In Clare Dixon and Marcelo Finger, editors, *Frontiers of Combining Systems*, volume 10483 of *Lecture Notes in Computer Science*, pages 172–188. Springer International Publishing, 2017.

[7] Anupam Das. On the logical complexity of cyclic arithmetic. *Logical Methods in Computer Science*, Volume 16, Issue 1, January 2020.

[8] M. Echenim and N. Peltier. Combining induction and saturation-based theorem proving. *Journal of Automated Reasoning*, pages 1–42, Mar 2019.

[9] Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*, volume 3. Cambridge University Press, 2017.

[10] Stefan Hetzl and Tin Lok Wong. Some observations on the logical foundations of inductive theorem proving. *Logical Methods in Computer Science*, Volume 13, Issue 4, April 2018.

[11] Abdelkader Kersani and Nicolas Peltier. Combining superposition and induction: A practical realization. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 7–22. Springer Berlin Heidelberg, 2013.

[12] Robert Nieuwenhuis and Albert Rubio. Chapter 7 - paramodulation-based theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, Handbook of Automated Reasoning, pages 371 – 443. North-Holland, Amsterdam, 2001.

[13] J. R. Shoenfield. Open sentences and the induction axiom. *Journal of Symbolic Logic*, 23(1):7–12, 1958.

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