Archive for Mathematical Logic (2024) 63:813–835
https://doi.org/10.1007/s00153-024-00923-8

Mathematical Logic

Quantifier-free induction for lists

Stefan Hetzl[1] · **Jannik Vierling**[1]

Received: 27 April 2023 / Accepted: 21 March 2024 / Published online: 20 April 2024

Abstract
We investigate quantifier-free induction for Lisp-like lists constructed inductively from the empty list *nil* and the operation *cons*, that adds an element to the front of a list. First we show that, for $m \geq 1$, quantifier-free m-step induction does not simulate quantifier-free $(m + 1)$-step induction. Secondly, we show that for all $m \geq 1$, quantifier-free m-step induction does not prove the right cancellation property of the concatenation operation on lists defined by left-recursion.

Keywords Weak theories of arithmetic · Theories of lists · Automated inductive theorem proving · Transfinite lists

Mathematics Subject Classification 03C62 · 03F30 · 03H15 · 68V15 · 03B35 · 03B70

1 Introduction

In this article we consider Lisp-like lists in the context of the automation of proof by mathematical induction. The subject of automated inductive theorem proving (AITP) aims at automating the process of proving statements about inductively constructed objects such as natural numbers, lists and trees. The formal verification of software is a particularly prominent application of automated inductive theorem proving. Since every non-trivial program contains loops or recursion, some form of mathematical induction is necessary to reason about such programs. By Gödel's incompleteness theorem the task addressed by AITP is in general not even semi-decidable. Therefore, there is a lot more freedom in the choice of the proof systems than in the case of first-order validity. For that reason and because of technical constraints, a great variety of methods have been developed for that purpose. To name just a few examples, there

✉ Stefan Hetzl
stefan.hetzl@tuwien.ac.at

Jannik Vierling
jannik.vierling@tuwien.ac.at

[1] Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Vienna, Austria

are methods based on recursion analysis [5], integration into saturation-based provers [10, 21, 25], cyclic proofs [3], theory exploration [6], proof by consistency [8].

The current methodology in automated inductive theorem proving concentrates primarily on the implementation of systems and their empirical evaluation. The work in this article is part of a research program that aims at complementing this state of the art by focusing on the formal analysis of methods for automated inductive theorem proving. In particular, we aim at understanding the theoretical limits of systems by developing upper bounds on the logical strength of methods. Establishing sufficiently tight upper bounds on the strength of AITP systems often allows us to provide practically meaningful unprovability results whereas an empirical evaluation only shows the failure of a particular implementation. Moreover, upper bounds typically reveal the particular form of induction underlying the AITP systems. This knowledge permits the direct comparison of methods and helps in judging the applicability of AITP systems to certain domains.

So far the work in this research program [17–20, 31] has concentrated on induction for natural numbers only. However, since lists and other inductive data types are fundamental structures of computer science, it is of paramount importance for the subject of AITP to analyze the mechanical properties of these inductive datatypes. In this article we make a first important step towards extending this research program to inductively defined lists. In particular, we show that the right cancellation property of the concatenation of lists is not provable by a form of induction used in some automated inductive theorem proving systems. With this result we pave the way for obtaining further unprovability results for AITP systems on lists and other inductive data types.

In the following we briefly mention some aspects of axiomatic theories of finite lists have been studied in theoretical computer science. In [22] an axiomatic theory of linear lists (Lisp-like lists) is defined and some basic results about consistency, completeness, and independency of the axioms are shown. Similar theories are considered in a more general setting in [23]. In [1, 2, 12] the computability aspects of list structures are investigated.

Axiomatic theories of lists are closely related to theories of concatenation studied in logic [24, 29]. Theories of concatenation axiomatise strings of symbols over a finite alphabet. Theories of concatenation have been proposed as alternative basic systems for the development of metamathematical results such as Gödel's incompleteness theorems and computability [9, 13, 14, 30, 32]. In such theories there is no need to develop a coding of finite sequences [13, 24]. Hence, theories of concatenation permit a more natural development of syntax.

In this article we consider the provability of the right-cancellation of the concatenation of finite lists from quantifier-free big-step first-order induction for Lisp-like lists. After recalling some basic concepts and notations in Sect. 2, we show the two main results of this article in Sects. 3 and 4. First, in Sect. 3, we show that in general m-step quantifier-free induction does not prove $(m + 1)$-step quantifier-free induction. This results sets induction on lists in contrast with induction for natural numbers where big-step quantifier-free induction is not stronger than one-step quantifier-free induction. Secondly, in Sect. 4, we show that for all $m \geq 1$, m-step induction, over the language consisting of the list constructors and a concatenation operator, does

not prove the right cancellation property of the concatenation operation. In order to show these unprovability results we will construct models whose domain contains sequences of transfinite length.

2 Preliminaries

In this section we introduce some concepts, notations, and results that we will use throughout the article. In Sect. 2.1 we recall some basic concepts and notations of many-sorted first-order logic. Section 2.2 defines some basic axioms of the list constructors and the traditional induction schema for lists as well as related terminology. Finally, in Sect. 2.3 we introduce some concepts on transfinite sequences, which we will use in the model theoretic constructions of Sects. 3 and 4.

2.1 Many-sorted first-order logic

We work in the setting of classical many-sorted first-order logic with equality. Let S be a finite set of sorts, then for each sort $s \in S$ we let $\mathfrak{V}_s$ be a countably infinite set of variable symbols of the sort s. We write $x : s$ to indicate that x is a variable symbol of sort s, that is, $x \in \mathfrak{V}_s$. When the sort of a variable is irrelevant or clear from the context, we omit the sort annotation and simply use the variable symbol. We assume that the sets of variable symbols for the sorts in S are pairwise disjoint. A many-sorted first-order language $\mathcal{L}$ over the sorts S is a set of predicate symbols of the form $P : s_1 \times \cdots \times s_n \to o$ and function symbols of the form $f : s_1 \times \cdots \times s_n \to s_{n+1}$, where P, f are symbols, $s_1, \ldots, s_n, s_{n+1} \in S$ and o is a special sort symbol assumed not to appear in S. For a function symbol f the expression $f : s_1 \times \cdots \times s_n \to s_{n+1}$ with $s_1, \ldots, s_{n+1} \in S$ indicates that f takes arguments of sorts s_1, …, s_n to a value of sort s_{n+1}. Similarly, for a predicate symbol P the expression of the form $P : s_1 \times \cdots \times s_n \to o$ indicates that P is a predicate with arguments of sorts s_1, …, s_n. Terms of $\mathcal{L}$ are constructed as usual from the variable symbols and function symbols according to their respective types. Each constructed term t has a uniquely determined sort s and, therefore, we call t an s-term. Formulas of $\mathcal{L}$ are constructed from terms, predicate symbols, the connectives $\top$, $\bot$, $\wedge$, $\neg$, $\vee$, $\to$ and the quantifiers $(\forall x : s)$, $(\exists x : s)$ for $s \in S$ and $x \in \mathfrak{V}_s$.

In this article we will make heavy, albeit elementary, use of model theoretic techniques. Hence, we recall some basic model theoretic concepts and notations. A first-order structure M for the language $\mathcal{L}$ (over sorts S) is a function that assigns: To each sort $s \in S$ a non-empty set $M(s)$; To each function symbol $f : s_1 \times \cdots \times s_n \to s_{n+1}$ a function $f^M : \times_{i=1}^n M(s_i) \to M(s_{n+1})$; To each predicate symbol $P : s_1 \times \cdots \times s_n \to o$ a set $P^M \subseteq \times_{i=1}^n M(s_i)$. A variable assignment σ is a function that assigns to each variable symbol $v : s$ with $s \in S$ an element of $M(s)$. We write $M, \sigma \models \varphi$ if the formula φ is true in M under the variable assignment σ. Let $\varphi(x_1 : s_1, \ldots, x_n : s_n, \vec{y})$ be a formula and $d_i \in M(s_i)$ for $i = 1, \ldots, n$, then we write $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models \varphi$ (or $M \models \varphi(d_1, \ldots, d_n, \vec{y})$) if $M, \sigma \models \varphi$, for all variable assignments σ with $\sigma(x_i) = d_i$ for $i = 1, \ldots, n$. Thus, in particular,

$M \models \varphi$ if $M, \sigma \models \varphi$ for all variable assignments σ. Let $t(x_1 : s_1, \ldots, x_n : s_n)$ be a term and $d_1, \ldots, d_n$ a finite sequence in $M(s_1) \times \cdots \times M(s_n)$, then we write $t^M(\vec{d})$ to denote the element b of M such that $M, \{x_i \mapsto d_i \mid i = 1, \ldots, n\} \models t = b$.

In the arguments given in Sects. 3 and 4 it is often necessary to consider terms and formulas of a language $\mathcal{L}$ under some partial variable assignment over an $\mathcal{L}$ structure M. In order to simplify the notation, we let $\mathcal{L}(M)$ denote the language $\mathcal{L}$ extended by a fresh function symbol $c_d : s$ for each element $d \in M(s)$ and sort $s \in S$. Moreover, we let the structure M interpret the language $\mathcal{L}(M)$ by letting M interpret c_d as the object d.

In this article we define a theory T to be a set of sentences, which we call the axioms of T. Let φ be formula, then we write $T \vdash \varphi$ if φ is provable in (many-sorted) first-order logic from the axioms of T. Let T_1, T_2 be theories, then $T_1 + T_2$ denotes the theory axiomatized by the set of sentences $T_1 \cup T_2$.

Finally, let us define some notation for some particular sets of formulas. By $\mathrm{Open}(\mathcal{L})$ we denote the set of quantifier-free formulas of the language $\mathcal{L}$. Let Φ be a set of formulas, then we write $\forall_1(\Phi)$ ($\exists_1(\Phi)$) for the set of formulas in Φ of the form $(\forall \vec{x})\varphi$ ($(\exists \vec{x})\varphi$) where φ is a quantifier-free formula and $\vec{x}$ is a possibly empty sequence of variables. We also write $\forall_1(\mathcal{L})$ for the formulas of the above form in the language $\mathcal{L}$.

2.2 Induction and lists

In this section we introduce the basic construction of finite Lisp-like lists that we work with in this article. We also recall the traditional induction schema for lists and its related terminology. Throughout the article we will consider various forms of induction that will be defined when needed. We use the traditional induction schema as defined in this section as a reference in the sense that we justify the other induction schemata in terms of the traditional one.

Now we will define the basic language of finite Lisp-like lists and the corresponding induction schema.

Definition 2.1 The language $\mathcal{L}_0$ consists of the sort i of elements and the sort list of finite lists. Moreover, the language $\mathcal{L}_0$ contains the function symbols $nil : \mathsf{list}$ and $cons : \mathsf{i} \times \mathsf{list} \to \mathsf{list}$.

Informally, the symbol nil denotes the empty list and $cons$ denotes the operation that adds a given element to the front of a given list. For the sake of legibility we will use upper case letters X, Y, Z and variants thereof to denote variables that range over the sort list. For these variables we omit the the sort annotation, that is, the "$: \mathsf{list}$" part.

The traditional induction schema for Lisp-like lists is analogous to the one for natural numbers with the exception that the induction step also quantifies over elements.

Definition 2.2 Let $\varphi(X, \vec{z})$ be a formula, then the formula $I_X\varphi$ is given by

$$(\varphi(nil, \vec{z}) \wedge (\forall X)(\forall x)(\varphi(X, \vec{z}) \to \varphi(cons(x, X), \vec{z}))) \to (\forall X)\varphi(X, \vec{z}).$$

For a set of formulas Φ, the theory Φ-IND is axiomatized by the universal closure of the formulas $I_X\varphi$, where $\varphi(X, \vec{z}) \in \Phi$.

The induction schema given above is parameterized by the set of possible induction formulas. This permits to consider various theories by varying the structure of the induction formulas.

We will also refer to the above induction principle as one-step induction, since the induction step proceeds by a step of size one. In Sect. 3 we will introduce the big-step induction principle that proceeds in larger steps.

When we work with theories of lists we usually work over the following base theory that provides the disjointness and the injectivity of the list constructors *nil* and *cons*.

Definition 2.3 The theory T_0 is axiomatized by the following axioms

$$nil \neq cons(x, X), \tag{L0.1}$$

$$cons(x, X) = cons(y, Y) \rightarrow x = y \wedge X = Y. \tag{L0.2}$$

2.3 Transfinite sequences

In this section we introduce some notations and definitions related to transfinite sequences, that is, sequences indexed by ordinals. Later on in Sects. 3 and 4, we will heavily rely on transfinite sequences, of length up to ω^3, for the construction of non-standard models of induction over lists.

Let $\mathcal{X}$ be a set and α be an ordinal number, then as usual $\mathcal{X}^\alpha$ denotes the set $\{f : \alpha \rightarrow \mathcal{X}\}$ of sequences of elements of $\mathcal{X}$ with length α. If $a \in \mathcal{X}^\alpha$ and $\beta < \alpha$ then we write a_β for $a(\beta)$, the element of a with index β. By $\mathcal{X}^{<\alpha}$ ($\mathcal{X}^{\leq\alpha}$) we denote the set $\bigcup_{\beta<\alpha} \mathcal{X}^\beta$ ($\bigcup_{\beta\leq\alpha} \mathcal{X}^\beta$). In particular, we denote by $\mathcal{X}^*$ the set $\mathcal{X}^{<\omega} = \bigcup_{i\in\mathbb{N}} \mathcal{X}^i$ of all finite sequences of elements of $\mathcal{X}$. Let $a \in \mathcal{X}^{\leq\alpha}$, then $|a|$ denotes the ordinal $\beta \leq \alpha$ such that $a \in \mathcal{X}^\beta$. The empty sequence ( ) (= $\varnothing$) is also denoted by ε and (x) denotes the one element sequence $\{0 \mapsto x\}$.

In the following we define concatenation of ordinal indexed sequences. The definition as given below relies on the well-definedness of ordinal subtraction of an ordinal β from an ordinal α when $\alpha \geq \beta$ (see [28, Theorem 8.8]).

Definition 2.4 Let $\mathcal{X}$ be a set, α, β ordinals, $a \in \mathcal{X}^\alpha$, and $b \in \mathcal{X}^\beta$, then the sequence $a \frown b \in \mathcal{X}^{\alpha+\beta}$ is defined by

$$(a \frown b)_\gamma := \begin{cases} a_\gamma & \text{if } \gamma < \alpha \\ b_\delta & \text{otherwise} \end{cases},$$

where $\gamma < \alpha + \beta$ and δ is the unique ordinal such that $\alpha + \delta = \gamma$.

Observe that the definition of concatenation of ordinal indexed sequences given above generalizes the concatenation of finite sequences, since

$$(a_0, \ldots, a_{n-1}) \frown (b_0, \ldots, b_{m-1}) = (a_0, \ldots, a_{n-1}, b_0, \ldots, b_{m-1}).$$

The concatenation of ordinal indexed sequences as defined above has some interesting properties.

Lemma 2.5 *Let $\mathcal{X}$ be a set, α, β, γ ordinals, $a \in \mathcal{X}^\alpha$, $b \in \mathcal{X}^\beta$, $c \in \mathcal{X}^\gamma$, then we have:*

(i) *associativity: $a \frown (b \frown c) = (a \frown b) \frown c$;*
(ii) *left cancellation: If $a \frown b = a \frown c$, then $b = c$.*

Proof For (i) let $\mu < \alpha + \beta + \delta$. By the associativity of ordinal addition we have $a \frown (b \frown c), (a \frown b) \frown c \in \mathcal{X}^{\alpha+\beta+\gamma}$. Now we have to consider three cases. If $\mu < \alpha$, then $\mu < \alpha + \beta$. Hence

$$(a \frown (b \frown c))_\mu = a_\mu = (a \frown b)_\mu = ((a \frown b) \frown c)_\mu.$$

If $\alpha \leq \mu < \alpha + \beta$, then there is a unique ordinal δ such that $\alpha + \delta = \mu$. Moreover, by the monotonicity properties of ordinal addition we have $\delta < \beta$. Hence

$$(a \frown (b \frown c))_\mu = (b \frown c)_\delta = b_\delta = (a \frown b)_\mu = ((a \frown b) \frown c)_\mu.$$

Finally, if $\alpha + \beta \leq \mu < \alpha + \beta + \gamma$, then there are unique δ_1 and δ_2 such that $\alpha + \delta_1 = \mu$ and $\alpha + \beta + \delta_2 = \mu$. Furthermore, we have $\delta_1 \geq \beta$, hence there is δ_3 such that $\beta + \delta_3 = \delta_1$. Thus $\mu = \alpha + \delta_1 = \alpha + \beta + \delta_3 = \alpha + \beta + \delta_2$, hence $\delta_2 = \delta_3$. Therefore

$$(a \frown (b \frown c))_\mu = (b \frown c)_{\beta+\delta_3} = c_{\delta_3} = c_{\delta_2} = ((a \frown b) \frown c)_\mu.$$

For (ii) observe that since $a \frown b \in \mathcal{X}^{\alpha+\beta}$, $a \frown c \in \mathcal{X}^{\alpha+\gamma}$ we have $\alpha + \beta = \alpha + \gamma$ and therefore by the left cancellation of ordinal addition $\beta = \gamma$. Now let $\delta < \beta$, then we have $(a \frown b)_{\alpha+\delta} = b_\delta = c_\delta = (a \frown c)_{\alpha+\delta}$. Hence, $b = c$. □

Observe, however, that since ordinal addition does not have right cancellation, the concatenation of ordinal indexed sequences does clearly also not have right cancellation.

For sequences we will often be interested in suffixes. In the following definition we introduce some notation for accessing the suffix of a sequence.

Definition 2.6 (*Sequence suffix*) Let $\mathcal{X}$ be a set, α, β ordinals with $\beta \leq \alpha$, and $a \in \mathcal{X}^\alpha$, then the sequence $a \upharpoonright \beta$ is given by

$$(a \upharpoonright \beta)_\gamma = a_{\beta+\gamma},$$

for $\gamma < \mu$ where μ is the unique ordinal such that $\beta + \mu = \alpha$.

Finally, let us give some notation for the sequence obtained by concatenating sequences of uniform length. This construction will be used in Sect. 4 and relies on ordinal division with remainder (see [28, Theorem 8.27]).

Definition 2.7 Let $\mathcal{X}$ be a set, α, β ordinals with $\alpha > 0$, and $\mathfrak{a} \in (\mathcal{X}^\alpha)^\beta$. The sequence $\lfloor \mathfrak{a} \rfloor \in \mathcal{X}^{\alpha \cdot \beta}$ is defined by

$$\lfloor \mathfrak{a} \rfloor_\xi := \mathfrak{a}_{\delta,\mu},$$

for $\xi < \alpha \cdot \beta$ where μ, δ are the unique ordinals such that $\xi = (\alpha \cdot \delta) + \mu$ with $\mu < \alpha$. Furthermore, for $a \in \mathcal{X}^\alpha$, we denote by a^β the sequence $\lfloor (a)_{\gamma<\beta} \rfloor$ consisting of β times the sequence a.

Example 2.8 Let $a = 0, 2, 4, 6, \ldots \in \mathbb{N}^\omega$ be the sequence of even numbers and $b = 1, 3, 5, 7, \ldots \in \mathbb{N}^\omega$ be the sequence of odd numbers. Then, e.g., $a_0 = 0$, $b_0 = 1$, and $b_1 = 3$. Let $\mathfrak{a} = (a, b) \in (\mathbb{N}^\omega)^2$. Then, in the notation of Definition 2.7, $\alpha = \omega$, $\beta = 2$, and $\lfloor \mathfrak{a} \rfloor = a \frown b = 0, 2, 4, 6, \ldots 1, 2, 5, 7, \ldots \in \mathbb{N}^{\omega \cdot 2}$. For, e.g., $\xi = \omega + 3$, we have $\delta = 1$ and $\mu = 3$, so $\lfloor \mathfrak{a} \rfloor_\xi = \mathfrak{a}_{\delta,\mu} = b_\mu = 7$. Moreover, $\lfloor \mathfrak{a} \rfloor \uparrow 2 = 4, 6, 8, \ldots 1, 3, 5, \ldots$ and $\lfloor \mathfrak{a} \rfloor \uparrow \omega = b$.

3 Big-step induction

Big-step induction is a generalization of the induction principle of Definition 2.2 in which the induction step proceeds by adding more than one element. Big-step induction and other induction principles are often used in automated inductive theorem provers [4]. Some formulas can be proved more naturally by a special induction principle. Hence a special induction principle may allow a prover to find a proof faster under the constraints of its proof search algorithm or even enable the prover to prove the formula in the first place [31]. It is therefore interesting to investigate the relation between the one-step induction principle and special induction principles implemented in AITP systems.

In this section we show the first main result of this article, namely that, for all $m \geq 1$, quantifier-free $(m+1)$-step induction for lists does not follow from quantifier-free m-step induction. In particular, quantifier-free big-step induction for lists cannot be reduced to quantifier-free one-step induction, which is in contrast to induction on natural numbers where such a reduction is possible (see for example [31]).

The definition below defines the big-step induction principle for lists considered in this article. Let us introduce some notation to make it easier to state big-step induction for lists. Let $t_1, \ldots, t_n$ be a possibly empty list of terms of sort i and T a term of sort list, then the term $cons(t_1, \ldots, t_n; T)$ is defined inductively by

$$\begin{aligned} &cons(; T) = T, \\ &cons(t_1, \ldots, t_{n+1}; T) = cons(t_1, \ldots, t_n, cons(t_{n+1}, T)). \end{aligned}$$

Definition 3.1 Let $\varphi(x, \vec{z})$ be a formula and $m \geq 1$, then the formula $I_{x\frown m}\varphi$ is given by

$$\left(\begin{array}{l} \bigwedge\limits_{i=1,\ldots,m} (\forall x_1, \ldots, x_{i-1})\varphi(cons(x_1, \ldots, x_{i-1}; nil), \vec{z}) \\ \wedge\ (\forall X)(\forall x_1, \ldots, x_m)(\varphi(X, \vec{z}) \to \varphi(cons(x_1, \ldots, x_m; X), \vec{z})) \end{array} \right) \to (\forall X)\varphi(X, \vec{z}).$$

Let Φ be a set of formulas and $m \geq 1$, then the m-step induction schema $\Phi\text{-IND}_{\frown m}$ over Φ is axiomatized by the universal closure of the formulas $I_{x\frown m}\varphi$ where $\varphi(x, \vec{z}) \in \Phi$.

A simple example of formulas that have natural proofs by big-step induction are the acyclicity formulas given below, which express that adding a finite number $n \geq 1$ of elements to a list results in a different list:

$$X \neq cons(x_1, \ldots, x_n; X). \tag{$\star$}$$

To prove this formula by n-step induction it suffices to proceed by induction on X in the formula itself. For the base case we have to show that $nil \neq cons(x_1, \ldots, x_n; nil)$, which follows readily from (L0.1). For the induction step we assume ($\star$). For a contradiction assume

$$cons(x'_1, \ldots, x'_n; X) = cons(x_1, \ldots, x_n, x'_1, \ldots, x'_n; X).$$

Then by an n-fold application of (L0.2) we obtain $x'_i = x_i$ for $i = 1, \ldots, n$ and

$$X = cons(x'_1, \ldots, x'_n; X).$$

This contradicts the induction hypothesis and thus completes the induction step. Interestingly, however, the acyclicity formula ($\star$) also has a slightly less natural proof using one-step quantifier-free induction.

Lemma 3.2 *The theory* $T_0 + \text{Open}(\mathcal{L}_0)\text{-IND}$ *proves*

(i) $X \neq cons(x_1, \ldots, x_n; X)$ *for* $n \geq 1$*;*
(ii) $X = nil \vee (\exists x')(\exists X')X = cons(x', X')$.

Proof For (i) we assume $X = cons(x_1, \ldots, x_n; X)$ and proceed by induction on X' in the formula

$$\underbrace{X' \neq X}_{\psi_1(X')} \wedge \underbrace{X' \neq cons(x_n, X)}_{\psi_2(X')} \wedge \cdots \wedge \underbrace{X' \neq cons(x_2, \ldots, x_n; X)}_{\psi_n(X')}.$$

For the base case we have to show $\psi_i(nil)$ for $i = 1, \ldots, n$. For $i > 1$, this follows easily from (L0.1) and for $i = 0$ we obtain $X' \neq X$ from the assumption $X = cons(x_1, \ldots, x_n; X)$ and (L0.1). For the induction step we assume $\bigwedge_{i=1}^{n} \psi_i(X')$.

Let $i \in \{1, \ldots, n\}$. If $i = 1$ assume $cons(x', X') = X$, then by the assumption $X = cons(x_1, \ldots, x_n; X)$ and (L0.2) we obtain $X' = cons(x_2, \ldots, x_n; X)$ which contradicts the assumption $\psi_n(X')$. If $i > 1$, then assume $cons(x', X') = cons(x_{n-i+2}, \ldots, x_n; X)$, then by (L0.2) we obtain $X' = cons(x_{n-i+1}, \ldots, x_n; X)$, which contradicts $\psi_{i-1}(X')$. Hence, we finally obtain $(\forall X')\big(\bigwedge_{i=1}^{n} \psi_i(X')\big)$. Thus, in particular, we have $\bigwedge_{i=1} \psi_i(cons(x_1, \ldots, x_n; X))$. Therefore, we obtain

$$cons(x_1, \ldots, x_n; X) \neq X,$$

which contradicts the first assumption.

For (*ii*) we proceed by induction on Y in the formula $X \neq Y$. For the base case we have to show $X \neq nil$. Assume $X = nil$, then we are done. For the induction step, we assume $X \neq Y$ and $X = cons(y, Y)$ and we are done. Hence we have $(\forall Y)X \neq Y$. Thus in particular $X \neq X$, which is a contradiction and thus implies the claim. □

This gives rise to the question whether a similar technique as we have used to prove the acyclicity formulas with quantifier-free induction is also possible in general. It is straightforward to see that we can simulate big-step induction with single-step induction by making use of universal quantifiers and conjunction. For the sake of completeness we recall the argument.

Lemma 3.3 *Let $m \geq 1$ and $\varphi(X, \vec{z})$ be a formula, then*

$$\vdash I_X \bigwedge_{i=1}^{m} (\forall x_1) \ldots (\forall x_{i-1}) \varphi(cons(x_1, \ldots, x_{i-1}; X), \vec{z}) \rightarrow I_{X \curvearrowright m} \varphi(X, \vec{z}).$$

Proof Assume $(\forall x_1) \ldots (\forall x_{i-1}) \varphi(cons(x_1, \ldots, x_{i-1}; nil), \vec{z})$ for $i = 1, \ldots, m$ and

$$(\forall X)(\forall x_1) \ldots (\forall x_m)(\varphi(X, \vec{z}) \rightarrow \varphi(cons(x_1, \ldots, x_m; X), \vec{z})). \tag{$\star$}$$

Clearly it suffices to show the following formula.

$$\bigwedge_{j=1}^{m} (\forall x_1) \ldots (\forall x_{j-1}) \varphi(cons(x_1, \ldots, x_{j-1}; X), \vec{z}). \tag{$\dagger$}$$

We proceed by induction on X in the formula ($\dagger$). The base case follows immediately from the assumptions. For the induction step case we assume ($\dagger$). Now let $i \in \{1, \ldots, m\}$, let $x', x_1, \ldots, x_{i-1}$ be fixed but arbitrary. If $i < m$, then we have to show $\varphi(cons(x_1, \ldots, x_{i-1}, x'; X), \vec{z})$, which follows from the induction hypothesis with $j = i + 1$. If $i = m$, then we have to show $\varphi(cons(x_1, \ldots, x_{m-1}, x'; X), \vec{z})$. By ($\dagger$) with $j = 1$, we have $\varphi(X, \vec{z})$, hence by ($\star$) we obtain the desired formula. □

Remark 3.4 When the domain of the elements provably consists of a finite number of elements $n \geq 1$, then the quantifiers over elements in the induction formula of Lemma 3.3 can be replaced by a conjunction. Hence, in this situation $(m + 1)$-step induction reduces to m-step induction without an increase in quantifier-complexity of the induction formulas.

However, as we will show, the increase of the quantifier complexity when simulating big-step induction with one-step induction is in general unavoidable. The remainder of the section is devoted to the proof of the following proposition.

Definition 3.5 The language $\mathcal{L}_A$ extends the base language of lists $\mathcal{L}_0$ by the predicate symbol $A : \mathsf{list} \to o$.

Proposition 3.6 *Let $m \geq 2$, then*

$$T_0 + \bigcup_{1 \leq j < m} \text{Open}(\mathcal{L}_A)\text{-IND}_{\frown j} \nvdash I_{x \frown m} A(x).$$

This proposition entails, in particular, that $T_0 + \text{Open}(\mathcal{L}_A)\text{-IND} \nvdash \text{Open}(\mathcal{L}_A)\text{-IND}_{\frown 2}$.

We will show the above claim by constructing a model of quantifier-free induction over the language $\mathcal{L}_A$ in which the predicate A does not satisfy two-step induction. In such a model we call an element a standard element if it can be expressed as a term of the form $cons(x_1, \ldots cons(x_n, nil))$ under a suitable variable assignment. All other elements are called the non-standard elements. By Lemma 3.2.(*ii*) a non-standard element can be decomposed any finite number of times and thus resemble transfinite sequences of length at least ω. The model constructed in Definition 3.8 will use transfinite sequences of length up to ω as the non-standard elements. Since for example the transfinite sequence v^ω with $v \in \mathbb{N}^*$ satisfies $v^\omega = v \frown v^\omega$ it violates the acyclicity property $X \neq cons(x_1, \ldots, x_{|v|}, X)$ (see Lemma 3.2). Hence we have to avoid sequences that absorb a finite prefix.

The following definition introduces the non-standard elements that we use for the model constructed in this section.

Definition 3.7 Let $k \in \mathbb{N}$, then by N_k we denote the sequence $(i)_{k \leq i < \omega}$. Now we define

$$\mathcal{N} \coloneqq \{w \frown N_k \mid w \in \mathbb{N}^*, k \in \mathbb{N}\}.$$

Let $N \in \mathcal{N}$, then there is a unique decomposition $N = w \frown N_k$ such that $|w|$ and k are minimal. We write w_N for this w and k_N for this k. We call w_N the main prefix of N and N_{k_N} the main suffix of N.

We can now define a structure whose domain consists of the finite sequences of natural numbers and the non-standard elements defined above.

Definition 3.8 Let $m \geq 1$, then the structure M_1^m interprets the sort i as the natural numbers and the sort list as $M_1^m(\mathsf{list}) = \mathbb{N}^* \cup \mathcal{N}$. Furthermore, M_1^m interprets the list constructors as $nil^{M_1^m} \coloneqq \varepsilon$ and $cons^{M_1^m}(n, l) \coloneqq (n) \frown l$ and the predicate symbol $A : \mathsf{list} \to o$ as

$$A^{M_1^m} \coloneqq \mathbb{N}^* \cup \left\{N \in \mathcal{N} \mid w_N \neq \varepsilon \text{ or } m \nmid k_N\right\}.$$

We say that an element l_1 is a predecessor of an element l_2 if there are $k, n_1, \ldots, n_k \in \mathbb{N}$ such that $M_1^m \models l_2 = cons(n_1, \ldots, n_k; l_1)$.

We start by observing that the structure defined above satisfies the basic axioms of the constructors *nil* and *cons* of finite sequences.

Lemma 3.9 *Let $m \geq 1$, then $M_1^m \models T_0$.*

Proof We start with the axiom (L0.1). We have $nil^{M_1^m} = \varepsilon = \varnothing$. Let $n \in \mathbb{N}$ and $l \in M_1^m(\mathsf{list})$, then $(0, n) \in (n) \frown l = cons^{M_1^m}(n, l)$. Hence, $nil^{M_1^m} \neq cons^{M_1^m}(n, l)$ and therefore $M_1^m \models$ (L0.1). Now let $n_1, n_2 \in \mathbb{N}$ and $l_1, l_2 \in M_1^m(\mathsf{list})$ and assume that $cons^{M_1^m}(n_1, l_1) = (n_1) \frown l_1 = (n_2) \frown l_2$. Clearly, $n_1 = n_2$, hence by Lemma 2.5 we immediately obtain $M_1^m \models$ (L0.2). □

The following lemma shows that unary A predicates of M_1^m, eventually periodically become true on predecessors of non-standard elements.

Lemma 3.10 *Let $m \geq 1$, $t(X)$ be a $\mathcal{L}_A(M_1^m)$ term, then there is a $K \in \mathbb{N}$ such that for all $k \in \mathbb{N}$ with $k \geq K$ and $m \nmid k$, $M_1^m \models A(t(N_k))$.*

Proof There clearly is a $w \in \mathbb{N}^*$ such that $M_1^m \models t(l) = w \frown l$ for all $l \in M_1^m(\mathsf{list})$. If $w = \varepsilon$, then we are done by letting $K = 0$. Otherwise, we let $K = (w)_{|w|-1} + 2$. For $k \geq K$, the sequence N_k is the main suffix of the sequence $w \frown N_k$ and the main prefix of $w \frown N_k$ is not empty. Thus $M_1^m \models A(t(N_k))$. □

Informally, the following lemma states that unary equational predicates over elements of M_1^m eventually stabilize.

Lemma 3.11 *Let $m \geq 1$ and $E(X)$ an $\mathcal{L}_A(M_1^m)$ equation. If $M_1^m \not\models E(X)$ then there exists $K \in \mathbb{N}$ such that firstly $M_1^m \not\models E(w)$ for all $w \in \mathbb{N}^*$ with $|w| \geq K$ and secondly $M_1^m \not\models E(N_k)$ for all $k \geq K$.*

Proof The case where $X \notin Var(E)$ is trivial. Let $E(X)$ be $u(X) = v(X)$. If $X \in Var(u)$ and $Var(v) = \varnothing$, then there is $w \in \mathbb{N}^*$ and $l' \in M_1^m(\mathsf{list})$ such that $M_1^m \models u(l) = w \frown l$ for all $l \in M_1^m(\mathsf{list})$ and $M_1^m \models v = l'$. If $|w| > |l'|$, then we are done by letting $K = 0$. Otherwise, if $|w| \leq |l'|$ we consider the prefix of l'. If w is not a prefix of l', then again we are done by letting $K = 0$. If w is the prefix of l', then $l' = w \frown l''$ for some $l'' \in \mathbb{N}^{\leq |l'|}$. We have $l'' \in M_1^m(\mathsf{list})$, since $M_1^m(\mathsf{list})$ is closed under predecessors. Thus $M_1^m \models E(l)$ if and only if $l = l''$. If l'' is a standard element, then $M_1^m \not\models E(l)$ for all $l \in M_1^m(\mathsf{list})$ with $|l| > |l''|$. Hence, we let $K = |l''| + 1$. Otherwise, if l'' is non-standard, then we readily have $M_1^m \not\models E(l)$ for all $l \in \mathbb{N}^*$. Furthermore, we have $M_1^m \not\models E(l)$ for all non-standard $l \in M_1^m(\mathsf{list})$ with $l_0 \neq l_0''$. Hence, it suffices to let $K = l_0'' + 1$.

Now let us consider the case where $Var(u) \cap Var(v) = \{X\}$. There exist $w, w' \in \mathbb{N}^*$ such that for all $l \in M_1^m(\mathsf{list})$, $M_1^m \models u(l) = w \frown l$ and $M_1^m \models v(l) = w' \frown l$. Moreover, by the assumption that $M_1^m \not\models E(X)$ we have $w \neq w'$. Hence, $M_1^m \not\models E(l)$ for all $l \in M_1^m(\mathsf{list})$. Thus, we let $K = 0$. □

We are now ready to show that the structure M_1^m satisfies quantifier-free j-step induction for $1 \leq j < m$ over the language consisting of the list constructors *nil*, *cons*, and the predicate symbol A.

Lemma 3.12 *Let $m \geq 2$, then $M_1^m \models \bigcup_{1 \leq j < m} \mathrm{Open}(\mathcal{L}_A)\text{-IND}_{\frown j}$.*

Proof Let $j \in \mathbb{N}$ with $1 \leq j < m$ and $\varphi(X)$ be a quantifier-free $\mathcal{L}_A(M_1^m)$ formula. Assume that

$$M_1^m \models \varphi(cons(x_1, \ldots, x_{i-1}; nil)), \tag{$*$}$$

for $i = 1, \ldots, j$ and

$$M_1^m \models \varphi(X) \to \varphi(cons(x_1, \ldots, x_j; X)), \tag{$\star$}$$

Let $l \in M_1^m(\mathsf{list})$. We have to show that $M_1^m \models \varphi(l)$. If l is standard, then we are done by a straightforward induction on $|l|$ making use of ($*$) and ($\star$).

Now let us consider the case where l is non-standard, that is, $l \in \mathcal{N}$. Let $E_1(X), \ldots, E_n(X)$ be all the list equations of φ with $M_1^m \not\models E_i(X)$ for $i = 1, \ldots, n$. Then by Lemma 3.11 there exists $K \in \mathbb{N}$ such that $M_1^m \not\models E_i(w)$ for all $w \in \mathbb{N}^*$ with $|w| \geq K$ and $M_1^m \not\models E_i(N_k)$ for all $k \geq K$.

Now let $A(t_1(X)), \ldots, A(t_p(X))$ be all the A atoms of φ. By Lemma 3.10, there exists $K' \geq K$ such that for all $k \in \mathbb{N}$ with $k \geq K'$ and $m \nmid k$, we have $M \models A(t_q(N_k))$ for $q = 1, \ldots, p$. Hence, by taking a sufficiently long prefix w of l ($|w| \geq j$), we obtain $K'' \geq K'$ such that $l = w \frown N_{K''}$ and $m \mid K'' - 1$. Since $m \mid K'' - 1$ and $j < m$, we have $m \nmid K'' + i$ for $i = 0, \ldots, j-1$. Thus, $M_1^m \models A(t_q(l \uparrow |w| - i))$ for $q = 1, \ldots, p$ and $i = 0, \ldots, j-1$.

Let $\psi(X)$ be any atom of $\varphi(X)$ and $w' \in \mathbb{N}^*$ with $|w'| \geq K$, then by the above, for $i = 0, \ldots, j-1$, we have $M_1^m \models \psi(w')$ if and only if $M_1^m \models \psi(l \uparrow |w| - i)$. Hence, $M_1^m \models \varphi(l \uparrow |w| - i) \leftrightarrow \varphi(w')$. In the first part of the proof we have already shown that $M_1^m \models \varphi(w')$. Hence, we have $M_1^m \models \varphi(l \uparrow |w| - i)$.

Therefore, by a straightforward induction starting with $M_1^m \models \varphi(l \uparrow |w| - i)$ for $i = 0, \ldots, j-1$ and by making use of ($\star$) we obtain $M_1^m \models \varphi(w' \frown (l \uparrow |w|))$ for all $w' \in \mathbb{N}^*$. In particular, we have $M_1^m \models \varphi(l)$. □

Lemma 3.13 *Let $m \geq 2$, then $M_1^m \not\models I_{x \frown m} A(x)$.*

Proof We have $M_1^m \models A(w)$ for all $w \in \mathbb{N}^*$, hence in particular

$$M_1^m \models A(cons(x_1, \ldots, x_{j-1}; nil)) \text{ for } j = 1, \ldots, m.$$

Now we consider the induction step. Let $l \in M_1^m(\mathsf{list})$ and $n_1, \ldots, n_m \in \mathbb{N}$. If $l \in \mathbb{N}^*$, then by the above we have

$$(cons(n_1, \ldots, n_m; l))^{M_1^m} \in \mathbb{N}^* \subseteq A^{M_1^m}.$$

Hence, $M_1^m \models A(X) \to A(cons(n_1, \ldots, n_m; l))$. For $l \in \mathcal{N}$ we show the contrapositive of the induction step. Suppose first that $(n_1, \ldots, n_m) \frown l \notin A^{M_1^m}$. Hence, we have $(n_1, \ldots, n_k) \frown l = N_k$ for some $k \in \mathbb{N}$ and $m \mid k$. Thus, $l = N_{k+m}$ that is $l \notin A^{M_1^m}$. Hence, $M_1^m \models A(x) \to A(cons(x_1, \ldots, x_m; X))$. However, we also have $N_0 \notin A^{M_1^m}$ because $w_{N_0} = \varepsilon$ and $k_{N_0} = 0$. Hence, $M_1^m \not\models I_{x \frown m} A(x)$. □

Proof of Proposition 3.6 An immediate consequence of Lemmas 3.9 and 3.12, and Lemma 3.13. □

So far we have shown that simulating quantifier-free $m+1$-step induction over lists with m-step induction, is not possible when induction formulas are quantifier-free. The simulation of big-step induction in Lemma 3.3 by one-step induction makes use of universal quantifiers and conjunction. This gives rise to the question whether the use of conjunction is necessary. We conjecture that it is necessary in the following sense. By Clause($\mathcal{L}$) we denote the set of all clauses (disjunctions of atoms and their negation) over the language $\mathcal{L}$.

Conjecture 3.14 *Let $m \geq 2$, then*

$$T_0 + \bigcup_{1 \leq j < m} \forall_1 \text{Clause}(\mathcal{L}_A)\text{-IND}_{\frown j} \nvdash I_{x \frown m} A(x).$$

This conjecture is particularly interesting for the methods presented in [15, 16, 25]. As shown in [19, 31] these methods carry out induction on literals and clauses. However, the results in [19, 31] are formulated for induction over natural numbers and need to be adapted to the case for induction over lists and other recursive datatypes. A positive answer to the conjecture above together with analogues of the results [19, 31] would provide a formal justification for the necessity to implement more powerful induction rules that handle conjunction and quantification such as described in [16]. As a byproduct, the formulas $I_{X \frown m} A(X)$ with $m \geq 1$ form a set of benchmark problems of increasing difficulty for automated theorem provers.

The above shows that mechanizing induction on lists is more complicated than induction on natural numbers in the sense that a reduction of big-step induction to one-step induction requires induction formulas with a higher quantifier-complexity. In the following we will consider lists with a concatenation operation and we will show that big-step induction does not prove the right cancellation of concatenation.

4 Right cancellation of list concatenation

In the previous section we have shown that quantifier-free $(m+1)$-big step induction is strictly stronger than quantifier-free m-step induction, but not stronger than $\forall_1$ induction. In this section we show that big-step quantifier-free induction is in general strictly weaker than $\forall_1$ induction. We will prove this result by showing that the right cancellation property of the append operation on lists can not be proved with quantifier-free big-step induction on lists. This result is of particular interest for the automation of proof by mathematical induction, since it implies the necessity to work with induction rules that exceed the power quantifier-free big-step induction to handle comparatively basic properties such as the right cancellation of list concatenation.

In the following we will work with a language that extends the base language of lists $\mathcal{L}_0$ by an infix symbol for the concatenation of lists. We will work with the usual left-recursive definition of concatenation.

Definition 4.1 The infix function symbol $\cdot \frown \cdot : \mathsf{list} \times \mathsf{list} \to \mathsf{list}$ represents the append operation on lists. We define the language $\mathcal{L}_1$ to be $\mathcal{L}_0 \cup \{\frown\}$. The theory T_1 extends the base theory of lists T_0 by the following axioms

$$nil \frown Y = Y, \tag{L1.1}$$

$$cons(x, X) \frown Y = cons(x, X \frown Y). \tag{L1.2}$$

In the following lemmas we prove several properties about lists, and in particular the concatenation operation, using increasingly powerful induction principles. We start by proving some simple properties with quantifier-free induction.

Lemma 4.2 *The theory* $T_1 + \mathrm{Open}(\mathcal{L}_1)$-IND *proves the following formulas*

(i) $X \frown nil = X$,
(ii) $X \frown (Y \frown Z) = (X \frown Y) \frown Z$.

Proof For both formulas we use a straightforward induction on X and making use of (L1.1), (L1.2). □

We prove the next property, the right cancellation for single-element lists, using simultaneous induction on two variables.

Definition 4.3 Let $\varphi(X, Y, \vec{z})$ be a formula, then the formula $I_{X,Y}\varphi$ is given by

$$\begin{pmatrix} (\forall X)\varphi(X, nil, \vec{z}) \wedge (\forall Y)\varphi(nil, Y, \vec{z}) \\ \wedge (\forall X)(\forall Y)(\forall x)(\forall y)(\varphi(X, Y, \vec{z}) \to \varphi(cons(x, X), cons(y, Y), \vec{z})) \end{pmatrix} \\ \to (\forall X)(\forall Y)\varphi(X, Y, \vec{z}).$$

Let Γ be a set of formulas, then the theory Γ-DIND is axiomatized by the sentences $(\forall \vec{z})I_{X,Y}\varphi(X, Y, \vec{z})$ with $\varphi(X, Y, \vec{z}) \in \Gamma$.

Lemma 4.4 $T_1 + \mathrm{Open}(\mathcal{L}_1)$-DIND *proves*

$$Y \frown cons(x, nil) = Z \frown cons(x, nil) \to Y = Z.$$

Proof We proceed by induction on Y and Z simultaneously. We consider only one of the two base cases, since the other one is symmetric. For the base case $Y = nil$ we assume $nil \frown cons(x, nil) = Z \frown cons(x, nil)$ and we have to show that $Z = nil$. First of all, by (L1.1) we obtain $cons(x, nil) = Z \frown cons(x, nil)$. By Lemma 3.2 we can consider two cases. If $Z = nil$, then we are done. Otherwise, there are z' and Z' such that $Z = cons(z', Z')$. Thus

$$\begin{aligned} cons(x, nil) &= cons(z', Z') \frown cons(x, nil) \\ &=_{(\text{L1.2})} cons(z', Z' \frown cons(x, nil)). \end{aligned}$$

Therefore, by (L0.2) we have in particular $nil = Z' \frown cons(x, nil)$. We apply Lemma 3.2 and consider two cases. If $Z' = nil$, then $nil = nil \frown cons(x, nil) = cons(x, nil)$, which contradicts (L0.1). Otherwise, there are z'' and Z'' such that $Z' = cons(z'', Z'')$, then, by (L1.2), $nil = cons(z'', Z'' \frown cons(x, nil))$, which contradicts (L0.1). For the induction step assume $Y \frown cons(x, nil) = Z \frown cons(x, nil) \to Y = Z$ and $cons(y, Y) \frown cons(x, nil) = cons(z, Z) \frown cons(x, nil)$. Then by (L1.2) and (L0.2) we obtain $y = z$ and

$$Y \frown cons(x, nil) = Z \frown cons(x, nil).$$

By the induction hypothesis we obtain $Y = Z$, thus, $cons(y, Y) = cons(z, Z)$. □

Observe that double induction is contained within induction on $\forall_1$ formulas when working modulo case analysis CA given by

$$(\forall X)\big(X = nil \vee (\exists X')(\exists x')X = cons(x', X')\big).$$

Lemma 4.5 $\mathrm{CA} + \forall_1(\mathcal{L})\text{-IND} \vdash \mathrm{Open}(\mathcal{L})\text{-DIND}$.

Proof Let $\varphi(X, Y, \vec{z})$ be a quantifier-free $\mathcal{L}$ formula. Let $X, Y, \vec{z}$ be fixed and assume $(\forall X)\varphi(X, nil, \vec{z})$, $(\forall Y)\varphi(nil, Y, \vec{z})$, and

$$(\forall X)(\forall Y)(\forall x)(\forall y)(\varphi(X, Y, \vec{z}) \to \varphi(cons(x, X), cons(y, Y), \vec{z})).$$

We proceed by induction on X in $(\forall Y)\varphi(X, Y, \vec{z})$. The base case follows immediately from the assumptions. For the step case assume $(\forall Y)\varphi(X, Y, \vec{z})$ and let Y be fixed. By CA we can consider two cases. If $Y = nil$, then we are done by the assumption. Otherwise, there are y' and Y' such that $Y = cons(y', Y')$. By the induction hypothesis, we obtain $\varphi(X, Y', \vec{z})$. Hence, by the third assumptions, we have $\varphi(cons(x, X), cons(y', Y'), \vec{z})$, that is, $\varphi(cons(x, X), Y, \vec{z})$. □

Using induction on a $\forall_1$ formula, we can straightforwardly prove the right cancellation of the append operation for arbitrary lists.

Lemma 4.6 *The theory* $T_1 + \forall_1(\mathcal{L}_1)\text{-IND}$ *proves*

$$Y \frown X = Z \frown X \to Y = Z.$$

Proof We proceed by induction on X in the formula

$$(\forall Y)(\forall Z)(Y \frown X = Z \frown X \to Y = Z).$$

For the base case, let Y and Z be arbitrary and assume $Y \frown nil = Z \frown nil$. By Lemma 4.2 we readily obtain $Y = Z$. For the step case we assume

$$(\forall Y)(\forall Z)Y \frown X = Z \frown X \to Y = Z.$$

and $Y \frown cons(x, X) = Z \frown cons(x, X)$. By (L1.1), (L1.2), and Lemma 4.2 we obtain

$$(Y \frown cons(x, nil)) \frown X = (Z \frown cons(x, nil)) \frown X.$$

By the induction hypothesis we obtain $Y \frown cons(x, nil) = Z \frown cons(x, nil)$. Hence, by Lemmas 4.4 and 4.5 we obtain $Y = Z$. □

In the remainder of this section we will show that right cancellation of append cannot be proved by quantifier-free big-step induction on lists.

Theorem 4.7

$$T_1 + \bigcup_{m \in \mathbb{N}} \text{Open}(\mathcal{L}_1)\text{-IND}_{\frown m+1} \nvdash Y \frown X = X \to Y = nil.$$

We proceed as usual by constructing a structure that satisfies the base theory of lists with append together with quantifier-free induction for lists, but which contains elements l_1, l_2 such that $l_1 \frown l_2 = l_2$ and $l_1 \neq \varepsilon$. Since the concatenation of transfinite sequences of length greater or equal to ω does not have the right cancellation property, as for example $a \frown a^\omega = a^\omega$, it seems natural to use concatenation as an interpretation of the append symbol $\frown$.

In Sect. 3 we have already mentioned that, in order to construct a model of $T_0 + \text{Open}(\mathcal{L}_0)$-IND we have to avoid transfinite sequences λ such that $\lambda = w \frown \lambda$ for some $w \in \mathbb{N}^*$, cf. Lemma 3.2. However, we may introduce sequences that have a transfinitely periodic structure, such as, the sequence $N_0^\omega = N_0 \frown N_0^\omega$ of length ω^2.

In the following we define the set of elements that we will use for the construction of the model of quantifier-free big-step induction.

Definition 4.8 The structure M_2 interprets the sort i as the set $\mathbb{N}$ and the sort list as the set $\mathfrak{L}$ given by

$$\left\{ \lfloor \mathfrak{l} \rfloor \frown w \mid w \in \mathbb{N}^*, \mathfrak{l} \in \mathcal{N}^\beta, \beta < \omega^2 \right\}.$$

Furthermore, the structure M_2 interprets the non-logical symbols as follows

$$\begin{aligned} nil^{M_2} &:= \varepsilon, \\ cons^{M_2}(n, l) &:= n \frown l, \\ l_1 \frown^{M_2} l_2 &:= l_1 \frown l_2. \end{aligned}$$

We will now first ensure that the structure M_2 defined above is indeed a well-defined $\mathcal{L}_1$ structure, that is, that it is closed under the functions nil^{M_2}, $cons^{M_2}$, and $\frown^{M_2}$.

Lemma 4.9 *M_2 is an $\mathcal{L}_1$ structure.*

Proof We have to show that M_2 is closed under the operations nil^{M_2}, $cons^{M_2}(\cdot,\cdot)$, and $\cdot \frown^{M_2} \cdot$. We have $nil^{M_2} = \varepsilon \in \mathbb{N}^* \subseteq \mathfrak{L}$. Now let $n \in \mathbb{N}$ and $l \in \mathfrak{L}$. Let $l = \lfloor (m_\gamma)_{\gamma \leq \beta} \rfloor \frown w$ with $\beta < \omega^2$, $m \in \mathcal{N}^\beta$, and $w \in \mathbb{N}^*$. If $\beta = 0$, then $n \frown l = n \frown \varepsilon \frown w = n \frown w \in \mathbb{N}^* \subseteq \mathfrak{L}$. Otherwise, if $0 < \beta$, then for $\gamma < \beta$ we let

$$m'_\gamma \coloneqq \begin{cases} (n) \frown m_0 & \text{if } \gamma = 0, \\ m_\gamma & \text{otherwise} \end{cases}$$

Now observe that $(n) \frown \lfloor (m_\gamma)_{\gamma<\beta} \rfloor = \lfloor (m'_\gamma)_{\gamma<\beta} \rfloor$ and clearly $m'_\gamma \in \mathcal{N}$, for all $\gamma < \beta$. Hence, $cons^{M_2}(n, l) \in \mathfrak{L}$. Now let $l_1, l_2 \in \mathfrak{L}$ and consider $l_1 \frown^{M_2} l_2$. If $l_1 \in \mathbb{N}^*$, then we use an analogous argument as above. If $l_2 \in \mathbb{N}^*$, then we clearly have $l_1 \frown l_2 \in \mathfrak{L}$. If l_1 and l_2 are non-standard, then for $i = 1, 2$ there are $\alpha_i < \omega^2$, $\mathfrak{a}_i \in \mathcal{N}^{\alpha_i}$, $w_i \in \mathbb{N}^*$ such that $l_i = \lfloor \mathfrak{a}_i \rfloor \frown w_i$. Moreover, there exists $\delta \leq \alpha_2$ and $w' \frown N_k \in \mathcal{N}$ such that $1 + \delta = \alpha_2$ and $l_2 = w' \frown N_k \frown \lfloor (\mathfrak{a}_{2,1+\gamma})_{\gamma<\delta} \rfloor$. Therefore, we have

$$l_1 \frown l_2 = \lfloor \mathfrak{a}_1 \rfloor \frown (w_1 \frown w' \frown N_k) \frown \lfloor (\mathfrak{a}_{2,1+\gamma})_{\gamma<\delta} \rfloor \frown w_2.$$

Since $w_1 \frown w' \frown N_k \in \mathcal{N}$ and $\alpha_1 + \alpha_2 < \omega^2$ we have $l_1 \frown l_2 \in \mathfrak{L}$. □

Next we show that M_2 satisfies the basic axioms of the list constructors *nil* and *cons*, as well as those of the append symbol.

Lemma 4.10 $M_2 \models T_1$.

Proof Let $n \in \mathbb{N}$ and $l \in \mathfrak{L}$, then, since every element of $\mathcal{N}$ has length ω, there is some ordinal $\alpha < \omega^3$ such that $l \in \mathbb{N}^\alpha$. Hence, $cons^{M_2}(n, l) \in \mathbb{N}^{1+\alpha}$. Therefore $cons^{M_2}(n, l) \neq nil^{M_2} = \varepsilon = \varnothing$. Thus $M_2 \models$ (L0.1). Now let $n_1, n_2 \in \mathbb{N}$ and $l_1, l_2 \in \mathfrak{L}$ and assume that $n_1 \frown l_1 = n_2 \frown l_2$. For $i = 1, 2$, let $\alpha_i < \omega^3$ such that $l_i \in \mathbb{N}^{\alpha_i}$. We thus have $1 + \alpha_1 = 1 + \alpha_2$ which implies $\alpha_1 = \alpha_2$. Therefore, $n_1 = (n_1 \frown l_1)_0 = (n_2 \frown l_2)_0 = n_2$. Let $\gamma < \alpha_1$, then $l_{1,\gamma} = (n_1 \frown l_1)_{1+\gamma} = (n_2 \frown l_2)_{1+\gamma} = l_{2,\gamma}$. Thus, $l_1 = l_2$. Hence $M_2 \models$ (L0.2). Now let $l \in \mathfrak{L}$. We have $nil^{M_2} \frown l = \varepsilon \frown l = l$. Hence, $M_2 \models$ (L1.1). Now let $n \in \mathbb{N}, l, l' \in \mathfrak{L}$. Then we have

$$cons^{M_2}(n, l) \frown^{M_2} l' = ((n) \frown l) \frown l' = (n) \frown (l \frown l') = cons^{M_2}(n, l \frown^{M_2} l').$$

Thus, $M_2 \models$ (L1.2). □

Since the domain of M_2 interprets the sort of lists as transfinite sequences and the append operation as the concatenation of transfinite sequences, we can decompose list terms as follows.

Lemma 4.11 *Let $t(X, \vec{y})$ be a $\mathcal{L}_1$ list-term and $\vec{b}$ elements of M_2, then there exist $n \in \mathbb{N}$ and $l_0, \ldots, l_n \in \mathfrak{L}$ such that*

$$M_2 \models t(X, \vec{b}) = l_0 \frown X \frown l_1 \frown \cdots \frown l_{n-1} \frown X \frown l_n.$$

Proof We proceed by induction on the structure of the term t. If t is *nil*, then $M_2 \models t = \varepsilon$, and thus we are done. If t is the variable X, then we are done by letting $n = 0$ and $l_0 = \varepsilon \in \mathbb{N}^0$. If t is of the form $cons(u, t')$, then $M_2 \models u(\vec{b}) = k$, for some $k \in \mathbb{N}$. Hence we apply the induction hypothesis in order to obtain $n' \in \mathbb{N}$ and $l'_0, \ldots, l'_{n'} \in \mathfrak{L}$ such that $M_2 \models t'(X, \vec{b}) = l'_0 \frown X \frown \cdots \frown l'_{n'-1} \frown X \frown l'_{n'}$. Hence,

$$M_2 \models t(X, \vec{b}) = (k) \frown l'_0 \frown X \frown \cdots \frown l'_{n'-1} \frown X \frown l'_{n'}.$$

Thus, we let $n = n'$ and $l_0 = (k) \frown l'_0$ and $l_i = l'_i$ for $1 \leq i \leq n$. If t is of the form $t_1 \frown t_2$, then simply apply the induction hypothesis to t_1 and t_2. □

Equational predicates over M_2 in one variable stabilize eventually in a similar way to Lemma 3.11.

Lemma 4.12 *Let $E(X)$ be an $\mathcal{L}_1(M_2)$ equation such that $M_2 \not\models E(X)$, then there exists $N \in \mathbb{N}$ such that $M_2 \not\models E((n) \frown l)$ for all $n \geq N$ and $l \in \mathfrak{L}$.*

Proof Let $E(X)$ be $t_1(X) = t_2(X)$, then by Lemma 4.11 for $i = 1, 2$ there exist $n_i \in \mathbb{N}$ and $l^i_0, \ldots, l^i_{n_i} \in \mathfrak{L}$ such that

$$M_2 \models t_i = l^i_0 \frown X \frown \cdots \frown l^i_{n_i-1} \frown X \frown l^i_{n_i}.$$

By the symmetry of equality we can assume $n_1 \leq n_2$ without loss of generality. Since $M_2 \not\models E(X)$ we either have $n_1 \neq n_2$ or $l^1_i \neq l^2_i$ for some $i \in \{0, \ldots, n_1\}$. We start by assuming that $l^1_i = l^2_i$ for $i = 0, \ldots, n_1$ and $n_1 < n_2$. Then by the left cancellation of $\frown$ we obtain

$$M_2 \models E(X) \leftrightarrow \varepsilon = X \frown l^2_{n_1+1} \frown \cdots \frown l^2_{n_2-1} \frown X \frown l^2_{n_2}.$$

Hence, we have $M_2 \not\models E((n) \frown l)$, for all $n \in \mathbb{N}$ and $l \in \mathfrak{L}$. So in this case it suffices to take $N = 0$. Now consider the case where there exists $j \in \{0, \ldots, n_1\}$ such that $l^1_j \neq l^2_j$ and let $j_0 \in \{0, \ldots, n_1\}$ be the least such number. There are sequences $l, l^{1'}_{j_0}$ and $l^{2'}_{j_0}$ such that $l^i_{j_0} = l \frown l^{i'}_{j_0}$ for $i = 1, 2$ and either $|l^{1'}_{j_0}| = 0$, $|l^{2'}_{j_0}| \geq 1$, or $|l^{1'}_{j_0}| \geq 1$, $|l^{2'}_{j_0}| = 0$, or $|l^{1'}_{j_0}| \geq 1$, $|l^{2'}_{j_0}| \geq 1$ and $(l^{1'}_{j_0})_0 \neq (l^{2'}_{j_0})_0$. Hence, by left cancellation of concatenation, we obtain

$$\begin{aligned} M_2 \models E(X) \leftrightarrow\ & l^{1'}_{j_0} \frown X \frown \cdots \frown l^1_{n_1-1} \frown X \frown l_{n_1} \\ & = l^{2'}_{j_0} \frown X \frown \cdots \frown l^2_{n_2-1} \frown X \frown l_{n_2}. \end{aligned}$$

If $l^{1'}_{j_0} = \varepsilon$ and $l^{2'}_{j_0} \neq \varepsilon$, then for $n \neq (l^2_{j_0})_0$, we have $M_2 \not\models E((n) \frown l)$ for all $l \in \mathfrak{L}$. So in this case we take $N = (l^2_{j_0})_0 + 1$ The case where $l^{1'}_{j_0} \neq \varepsilon$ and $l^{2'}_{j_0} = \varepsilon$ is symmetric. Finally, in the case that $l^{1'}_{j_0}, l^{2'}_{j_0} \neq \varepsilon$ with $(l^{1'}_{j_0})_0 \neq (l^{2'}_{j_0})_0$, we trivially have $M_2 \not\models E(l)$ for all $l \in \mathfrak{L}$. So it suffices to take $N = 0$. □

As an immediate consequence of the previous lemma, we obtain the following result, which essentially says that for a non-standard element λ a list-equation $E(X)$ can eventually be stabilized for predecessors of λ.

Lemma 4.13 *Let $E(X)$ be an $\mathcal{L}_1(M_2)$ equation such that $M_2 \not\models E(X)$ and $\lambda \in \mathfrak{L}\backslash\mathbb{N}^*$. Then there exists $N \in \mathbb{N}$ such that $M_2 \not\models E(\lambda \uparrow n)$ for all $n \geq N$.*

Proof First by applying Lemma 4.12 we obtain m_0 such that $M \not\models E((m) \frown l)$ for all $m \geq m_0$ and $l \in \mathfrak{L}$. Since $\lambda \notin \mathbb{N}^*$, there clearly is $n_0 \in \mathbb{N}$ such that $\lambda \uparrow n_0 = N_{m_0} \frown \lambda'$ for some $\lambda' \in \mathfrak{L}$. Since $(\lambda \uparrow n_0 + k)_0 = m_0 + k \geq m_0$ for $k \in \mathbb{N}$, we have $M_2 \not\models E(\lambda \uparrow n)$ for all $n \geq n_0$. □

The previous two lemmas show that the truth value of formulas in M_2 on non-standard elements eventually synchronizes with that on standard elements, when considering sufficiently distant predecessors.

Lemma 4.14 *Let $\varphi(X)$ be an open $\mathcal{L}_1(M_2)$ formula and $\lambda \in \mathfrak{L}$, then there exists $n_0 \in \mathbb{N}$ such that*

$$M_2 \models \varphi(\lambda \uparrow n) \leftrightarrow \varphi((n)),$$

for all $n \geq n_0$.

Proof Clearly, it suffices to consider the list-equations of φ, since the i-equations do not depend on the variable X. Let $E_1(X), \ldots, E_k(X)$ be the atoms of φ with $M_2 \not\models E_i(X)$, for $i = 1, \ldots, k$. Then by Lemmas 4.13 and 4.12 there is $n_0 \in \mathbb{N}$ such that $M_2 \not\models E_i(\lambda \uparrow n)$ and $M_2 \not\models E_i((n))$ for $n \geq n_0$ and $i = 1, \ldots, k$. Since we have $M_2 \models E(X)$ for the other list-atoms of φ, we obtain $M_2 \models \varphi(\lambda \uparrow n) \leftrightarrow \varphi((n))$ for $n \geq n_0$. □

We are now ready to show that M_2 satisfies open big-step induction.

Proposition 4.15 *Let $m \in \mathbb{N}$ with $m \geq 1$, then $M_2 \models \mathrm{Open}(\mathcal{L}_1)\text{-IND}_{\frown m}$.*

Proof Let $\varphi(X)$ be a quantifier-free $\mathcal{L}_1(M_2)$ formula. Assume that

$$M_2 \models \bigwedge_{i=1,\ldots,m} \varphi(cons(x_1, \ldots, x_{i-1}; nil)), \tag{$*$}$$

$$M_2 \models \varphi(X) \to \varphi(cons(x_1, \ldots, x_m; X)). \tag{$\star$}$$

Let $\lambda \in \mathfrak{L}$. If $\lambda \in \mathbb{N}^*$, then a straightforward induction making use of ($*$) and ($\star$) yields $M_2 \models \varphi(\lambda)$. Now we consider the case $\lambda \notin \mathbb{N}^*$, that is, λ is a non-standard element. By Lemma 4.14 there is $n_0 \in \mathbb{N}$ such that $M_2 \models \varphi(\lambda \uparrow n)$ if and only if $M_2 \models \varphi((n))$ for all $n \geq n_0$. In particular, we thus have

$$M_2 \models \varphi(\lambda \uparrow n_0 + m + i) \leftrightarrow \varphi((n_0 + m + i))$$

for $i = 0, \ldots, m - 1$. Since, $M_2 \models \varphi(w)$ for all $w \in \mathbb{N}^*$, we obtain $M_2 \models \varphi(\lambda \uparrow n_0 + m + i)$ for $i = 0, \ldots, m - 1$. By a straightforward induction starting with

$M_2 \models \varphi(\lambda \uparrow n_0 + m - 1)$, ..., $M_2 \models \varphi(\lambda \uparrow n_0)$ and making use of ($\star$) we obtain $M_2 \models \varphi(w \frown (\lambda \uparrow n_0))$ for all $w \in \mathbb{N}^*$. Therefore, we have in particular $M_2 \models \varphi(\lambda)$. □

Proof of Theorem 4.7 Clearly, $N_0 \in \mathfrak{L}$. Since $N_0^\omega = \lfloor (N_0)_{\gamma<\omega} \rfloor$, we have $N_0^\omega \in \mathfrak{L}$. Now observe that $N_0 \frown N_0^\omega = N_0^\omega$ but $N_0 \neq \varnothing$. Hence, by Proposition 4.15 we are done. □

This result is of interest for automated inductive theorem proving, because it essentially provides a lower bound on the power necessary for the proof of a rather simple yet practically relevant property about the important datatype of lists.

The unprovability of right cancellation of concatenation is a first step towards a classification of the inductive power needed to prove certain practically interesting properties of finite Lisp-like lists. Theorem 4.7 as well as the auxiliary results of this section give rise to many related questions and conjectures that we will briefly discuss in the following.

We conjecture that even quantifier-free simultaneous induction on several variables with big-steps does not prove right cancellation of the concatenation operation. Let $\vec{x} = (x_1, \ldots, x_n)$ be a finite sequence and $i \in \mathbb{N}$ such that $1 \leq i \leq n$, then by $\vec{x}_{<i}$ we denote the sequence $(x_1, \ldots, x_{i-1})$. Similarly, $\vec{x}_{>i}$ denotes the sequence $(x_{i+1}, \ldots, x_n)$.

Definition 4.16 Let $\vec{X} = (X_1, \ldots, X_m)$ be pairwise distinct variables with $m \geq 1$, $\vec{p} = (p_1, \ldots, p_m)$ a sequence of non-zero natural numbers, and $\varphi(\vec{X}, \vec{z})$ a formula. The multivariate big-step list induction axiom $I^{\mathsf{list}}_{\vec{X} \frown \vec{p}} \varphi$ for φ is given by

$$\left(\begin{array}{l} \displaystyle\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{p_i} (\forall \vec{X}_{<i})(\forall \vec{X}_{>i})(\forall x_1, \ldots, x_{j-1}) \varphi(\vec{X}_{<i}, cons(x_1, \ldots, x_{j-1}; nil), \vec{X}_{>i}, \vec{z}) \\ \wedge (\forall \vec{X})(\forall \vec{x}^{\,p_1}) \ldots (\forall \vec{x}^{\,p_m}) \Big(\varphi(\vec{X}, \vec{z}) \to \varphi(cons(\vec{x}^{\,p_1}; X_1), \ldots, cons(\vec{x}^{\,p_m}; X_m), \vec{z}) \Big) \end{array} \right) \to (\forall \vec{X}) \varphi(\vec{X}).$$

where the $\vec{x}^{\,p_i}$ with $i \in \{1, \ldots, m\}$ are vectors of variables of sort i whose elements are all pairwise distinct. Let Φ be a set of formulas, then theory $\Phi\text{-IND}^{\mathsf{list}}_{\nearrow \frown}$ is axiomatized by $I^{\mathsf{list}}_{\vec{X} \frown \vec{p}} \varphi$ with $\varphi(\vec{X}, \vec{z}) \in \Phi$ and $\vec{X}$, $\vec{p}$ as above.

Conjecture 4.17 $T_1 + \text{Open}(\mathcal{L}_1)\text{-IND}^{list}_{\nearrow \frown} \nvdash Y \frown X = Z \frown X \to Y = Z$.

A positive answer to this question would thus greatly improve upon our Theorem 4.7. A related question of interest is whether single-element right cancellation can be proven by quantifier-free big-step induction in one variable.

The subject of AITP mainly focuses on the mechanization of induction in general, rather than on the mechanization of individual theories. Nevertheless, the theories of lists with concatenation considered in this section are of some practical relevance. Hence, it may be valuable to investigate their mechanization separately. Because of the homomorphic relation between natural numbers with addition and lists with concatenation, it could be especially interesting to investigate whether simple theories of

lists such as $T_1 + \forall_1(\mathcal{L}_1)$-IND have finite axiomatizations analogous to the one shown in [27] for natural numbers with addition.

Finally, let us observe that as an immediate consequence of Proposition 4.15 we obtain the unprovability of right-decomposition of list by open big-step induction.

Corollary 4.18 $T_1 + \bigcup_{m \geq 1} \text{Open}(\mathcal{L}_1)\text{-IND}_{\frown m}$ *does not prove*

$$X = nil \vee (\exists x')(\exists X')X = X' \frown cons(x', nil).$$

Proof Consider the element $N_0 \in M_2(\mathsf{list})$ and observe that $N_0 \neq nil$ but since $|N_0| = \omega$, we cannot express N_0 as $\lambda \frown (n)$ with $\lambda \in \mathbb{N}^{\leq \omega}$ and $n \in \mathbb{N}$. Now, the claim follows from Proposition 4.15. □

Clearly, the formula $X = nil \vee (\exists x')(\exists X')X = X' \frown cons(x', nil)$ is provable by induction on the formula itself, that is, by $\exists_1$ induction. This gives rise to the question whether right-decomposition can be proved by $\forall_1$ induction and more generally to the more general question how $\exists_1$ induction and $\forall_1$ induction over lists with concatenation are related. This question is relevant for AITP, since there are systems such as [21] that are based on $\exists_1$ induction [18] and systems such as [10] that are based on $\forall_1$ induction [31, Chapter 5]. We plan to investigate this question separately in the future.

5 Conclusion

In this article we have shown two main results about induction for lists. Firstly, in Sect. 3 we have shown that quantifier-free $(m + 1)$-step induction can in general not be simulated with quantifier-free m-step induction. In particular, this result thus renders impossible a reductive implementation of quantifier-free big-step induction in AITP systems with an induction mechanism based on quantifier-free induction. This observation may be relevant for future extensions of systems based on quantifier-free one-step induction mechanism, such as the AITP system described in [25, Section 3.2]. The idea is that whenever an induction principle can be reduced to a simpler one, then for the sake of soundness one should consider the reduction.

The second main result of this article, shown in Sect. 4, is the unprovability of right cancellation of the concatenation for lists by quantifier-free big-step induction. Thus automated inductive theorem provers have to implement a comparatively strong induction mechanism in order to the prove seemingly simple property of right cancellation of concatenation.

In the light of the results of Sect. 3, a natural choice would be to implement an induction principle that can handle at least $\forall_1$ induction formulas with conjunction. Such an induction principle permits a reductive implementation of $\forall_1$ big-step induction. An example of a system implementing such an induction mechanism is the one described in [10] and analyzed in [31, Chapter 5].

One direction for future research is to carry out similar investigations focusing on other datatypes, induction principles, and properties. In principle questions such as the one addressed in Sect. 4 could be considered for every problem in benchmark suites

such as [7] in order to obtain a classification of the difficulty of the problems that complements empirical results.

Furthermore, the results in this article raise a number of questions and conjectures that we would like to address in the future. In particular, we would like to investigate Conjecture 4.17, since a positive answer, showing that quantifier-free induction combining, both, simultaneous induction and big-step induction does not prove right cancellation of concatenation, would significantly strengthen the result of Sect. 4. Another interesting question is whether the right injectivity of concatenation (see Lemma 4.4) can be proved with quantifier-free big-step induction. Finally, the use of transfinite lists used in this article are reminiscent of streams defined by coinduction. It could be interesting investigate to which extent the techniques employed for the analysis of AITP systems can be transferred to systems that automate the coinduction principle such as [11, 26].

Acknowledgements The authors would like to thank the anonymous reviewer for many helpful comments that led to an improvement of the presentation of this paper.

Funding Open access funding provided by TU Wien (TUW).

References

1. Aleksandrova, S.A., Bazhenov, N.A.: On decidability of list structures. Sib. Math. J. **60**(3), 377–388 (2019)
2. Bazhenov, N.A.: Automatic structures and the theory of lists. Sib. Electron. Math. Rep. **12**, 714–722 (2015)
3. Brotherston, J., Gorogiannis, N., Petersen, R.L.: A generic cyclic theorem prover. In: Jhala, R., Igarashi, A. (eds.) Programming Languages and Systems, volume 7705 of Lecture Notes in Computer Science, pp. 350–367. Springer, Berlin (2012)
4. Bundy, A., Basin, D.A., Hutter, D., Ireland, A.: Rippling—Meta-Level Guidance for Mathematical Reasoning, Volume 56 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2005)
5. Bundy, A., van Harmelen, F., Hesketh, J., Smaill, A., Stevens, A.: A rational reconstruction and extension of recursion analysis. In: Sridharan, N.S. (ed.) Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, vol. 1, pp. 359–365. Morgan Kaufmann, Burlington (1989)
6. Claessen, K., Johansson, M., Rosén, D., Smallbone, N.: Automating inductive proofs using theory exploration. In: Bonacina, M.P. (ed.) Automated Deduction—CADE-24, Volume 7898 of Lecture Notes in Computer Science, pp. 392–406. Springer, Berlin (2013)
7. Claessen, K., Johansson, M., Rosén, D., Smallbone, N.: TIP: Tons of inductive problems. In: Kerber, M., Carette, J., Kaliszyk, C., Rabe, F., Sorge, V. (eds.) Intelligent Computer Mathematics, Volume 9150 of Lecture Notes in Computer Science, pp. 333–337. Springer, Berlin (2015)
8. Comon, H.: Inductionless induction. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, chapter 14, pp. 913–962. North-Holland, Amsterdam (2001)
9. Corcoran, J., Frank, W., Maloney, M.: String theory. J. Symb. Logic **39**(4), 625–637 (1974)

10. Cruanes, S.: Superposition with structural induction. In: Dixon, C., Finger, M. (eds.) Frontiers of Combining Systems, Volume 10483 of Lecture Notes in Computer Science, pp. 172–188. Springer, Berlin (2017)
11. Einarsdóttir, S.H., Johansson, M., Pohjola, J.Å.: Into the infinite—theory exploration for coinduction. In: Fleuriot, J.D., Wang, D., Calmet, J. (eds.) Artificial Intelligence and Symbolic Computation, volume 11110 of Lecture Notes in Computer Science, pp. 70–86. Springer, Berlin (2018)
12. Goncharov, S.S.: A theory of lists and its models. Vychislitel'nye Sistemy **114**, 84–95 (1986)
13. Grzegorczyk, A.: Undecidability without arithmetization. Stud. Logica **79**(2), 163–230 (2005)
14. Grzegorczyk, A., Zdanowski, K.: Undecidability and Concatenation. In: Ehrenfeucht, A., Marek, V.W., Srebrny, M. (eds.) Andrzej Mostowski and Foundational Studies, pp. 72–91. IOS Press, Amsterdam (2008)
15. Hajdú, M., Hozzová, P., Kovács, L., Schoisswohl, J., Voronkov, A.: Induction with generalization in superposition reasoning. In: Benzmüller, C., Miller, B.R. (eds.) Intelligent Computer Mathematics, volume 12236 of Lecture Notes in Computer Science, pp. 123–137. Springer, Berlin (2020)
16. Hajdu, M., Hozzová, P., Kovács, L., Voronkov, A.: Induction with recursive definitions in superposition. In: Piskac, R., Whalen, M.W. (eds.) Proceedings of the 21st Conference on Formal Methods in Computer-Aided Design – FMCAD 2021, volume 2 of Conference Series: Formal Methods in Computer-Aided Design, pp. 246–255. TU Wien Academic Press, Vienna (2021)
17. Hetzl, S., Vierling, J.: Clause Set Cycles and Induction. Log. Methods Comput. Sci. **16**(4), 11:1-11:17 (2020)
18. Hetzl, S., Vierling, J.: Unprovability results for clause set cycles. Theor. Comput. Sci. (2022)
19. Hetzl, S., Vierling, J.: Induction and Skolemization in saturation theorem proving. Ann. Pure Appl. Logic **174**(1) (2023)
20. Hetzl, S., Wong, T.L.: Some observations on the logical foundations of inductive theorem proving. Log. Methods Comput. Sci. **13**(4), 10:1-10:26 (2018)
21. Kersani, A., Peltier, N.: Combining superposition and induction: a practical realization. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) Frontiers of Combining Systems, volume 8152 of Lecture Notes in Computer Science, pp. 7–22. Springer, Berlin (2013)
22. Moore, D.J., Russell, B.: Axiomatic data type specifications: a first order theory of linear lists. Acta Informatica **15**, 193–207 (1981)
23. Oppen, D.C.: Reasoning about recursively defined data structures. In: Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL '78, pp. 151–157. Association for Computing Machinery, New York, NY, USA (1978)
24. Quine, W.V.: Concatenation as a basis for arithmetic. J. Symb. Logic **11**(4), 105–114 (1946)
25. Reger, G., Voronkov, A.: Induction in saturation-based proof search. In: Fontaine, P. (ed.), Automated Deduction—CADE 27, volume 11716 of Lecture Notes in Computer Science, pp. 477–494. Springer, Berlin (2019)
26. Rustan, K., Leino, M., Moskal, M.: Co-induction simply—automatic co-inductive proofs in a program verifier. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods, volume 8442 of Lecture Notes in Computer Science, pp. 382–398. Springer, Berlin (2014)
27. Shoenfield, J.R.: Open sentences and the induction axiom. J. Symb. Log. **23**(1), 7–12 (1958)
28. Takeuti, G., Zaring, W.M.: Introduction to Axiomatic Set Theory, volume 1 of Graduate Texts in Mathematics. Springer, Berlin (1971)
29. Tarski, A.: Der Wahrheitsbegriff in den formalisierten Sprachen. Studia Philosophica **1**, 261–405 (1935)
30. Thatcher, J.W.: Decision problems for multiple successor arithmetics. J. Symb. Logic **31**(2), 182–190 (1966)
31. Vierling, J.: The limits of automated inductive theorem provers. Ph.D. thesis, Technische Universität Wien (2022)
32. Visser, A., Commas, G.: A study of sequentiality and concatenation. Notre Dame J. Formal Logic **50**(1), 61–85 (2009)