A criterion for showing the non-existence of straightforward induction proofs

Elijah Schulzki
TU Wien, Austria
e12207791@student.tuwien.ac.at

June 2025

1 Introduction

In the automation of inductive proving, an important question is whether a given correct statement can be proven by induction on itself. Lundstedt gave instances of formulas that do not allow such a straightforward induction proof in [1].

The aim of this seminar paper is to unify the proofs by Lundstedt into a general criterion, that can be applied to statements of arithmetic as well as statements concerning certain inductive data types.

We will first give the definitions necessary for the formalization of the above question in the context of inductive data types, then we will state and prove the criterion, and finally we will apply it to the examples provided by Lundstedt as well as examples using the inductive data type of lists.

2 Preliminaries

We will firstly make precise, what me mean by a proof of a formula by induction on itself:

Definition 2.1 Let L be a language with $0, s \in L$, T an L-theory and $\varphi(x)$ an L-formula. $\forall x \varphi(x)$ has a *straightforward induction proof* in T iff

$$T, \varphi(0) \wedge \forall x(\varphi(x) \to \varphi(s(x) \vdash \forall x \varphi(x)$$

We will work with theories of a form similar to $T = \mathrm{Th}_{L'}(\mathbb{N}) \cup \{D_f^0, D_f^s\}$. Here, $L = L' \cup \{f\}$ and all $c, g, R \in L'$ have fixed interpretations $c^{\mathbb{N}}, g^{\mathbb{N}}, R^{\mathbb{N}}$. $\mathrm{Th}_{L'}(\mathbb{N}) = \{\psi \ L'\text{-formula} : \mathbb{N}, c \mapsto c^{\mathbb{N}}, g \mapsto g^{\mathbb{N}}, R \mapsto R^{\mathbb{N}}, \cdots \vDash \psi\}$ and D_f^0, D_f^s are the base and step case of a recursive definition of f. For example, $D_f^0 \equiv f(0) = 0$ and $D_f^s \equiv f(s(x)) = f(x) + s(x)$ make up a recursive definition of $f(n) = \sum_{i=0}^{n} i$.

We now introduce a more general context, where recursive definition and proofs by induction are also possible: inductive data types.

We work in a many-sorted first-order logic with sorts $s_1, \ldots, s_n$. A definition of an inductive data type D on top of $s_1, \ldots, s_n$ is given by a finite set of constructors $c_1, \ldots, c_k$ where, for $i = 1, \ldots, k$, c_i is a function symbol of type $D^{n_i} \times \tau_{n_i+1} \times \cdots \times \tau_{m_i} \to D$ with $\tau_{n_i+1}, \ldots, \tau_{m_i} \in \{s_1, \ldots, s_n\}$. In reference to a data type, n_i will always denote the number of arguments of type D of the constructor c_i and m_i will always denote the total number of arguments. In order for such a set of constructors to be a valid definition of an inductive data type, there has to be an $i \in \{1, \ldots, k\}$ with $n_i = 0$. All c_i with $n_i = 0$ are called base constructors, all others are called step constructors.

The natural numbers form the inductive data type $\mathtt{Nat}$ with constructors $0 : \mathtt{Nat}^0 \to \mathtt{Nat}$ and $s : \mathtt{Nat}^1 \to \mathtt{Nat}$. Another example is the inductive data type $\mathtt{List}(s)$ of lists with elements from a sort s, which is discussed in Chapter 5.

Like the natural numbers, every inductive data type has a standard model. For D an inductive data type defined on top of sorts $s_1, \ldots, s_n$ with constructors $c_1, \ldots, c_k$ and $A_1, \ldots, A_n$ interpretations of sorts, the set of constructor terms of D with respect to $A_1, \ldots, A_n$ is written as $\mathbb{T}^D(A_1, \ldots, A_n)$ an is defined as the smallest set X such that for all $i \leq k$, for all $a_1, \ldots, a_{n_i} \in X$ and $a_j \in M_{l(j)}$ for $n_i+1 \leq j \leq m_j$ and $l(j)$ such that $\tau_i^j = s_{l(j)}$, $c_i(a_1, \ldots, a_{m_j}) \in X$.

Definition 2.2 Let $s_1, \ldots, s_n$ be sorts such that $s_{l+1}, \ldots, s_n$ are inductive data types with s_i defined on top of $s_1, \ldots, s_{i-1}$. Let $A_1, \ldots, A_l$ be sets. Then the *standard model* $\mathcal{S}$ of $s_1, \ldots, s_n$ with respect to $A_1, \ldots, A_l$ is defined by

$$s_i^{\mathcal{S}} = \begin{cases} A_i \text{ for } i = 1, \ldots, l \\ \mathbb{T}_i^s(s_1^{\mathcal{S}}, \ldots, s_{n-1}^{\mathcal{S}}) \text{ for } i = l+1, \ldots, n \end{cases}$$

and $c^{\mathcal{S}} = c$ for all constructors c.

Now we move on to recursive definitions:

Definition 2.3 Let D be an inductive data type defined over sorts $s_1, \ldots, s_n$ by constructors $c_1, \ldots, c_k$, let $f : D \times \sigma_1 \times \cdots \times \sigma_l \to \tilde{\sigma}$ be a function symbol with $\sigma_1, \ldots, \sigma_l, \tilde{\sigma} \in \{s_1, \ldots, s_n, D\}$, $f \notin L$. Then a *primitive recursive definition* of f consists of equations $D_f^{c_1}, \ldots, D_f^{c_k}$ with

$$f(c_i(x_1, \ldots, x_{m_i}), \bar{z}) = t_i(x_1, \ldots, x_{m_i}, f(x_1, \cdot), \ldots, f(x_{n_i}, \cdot), \bar{z}) \qquad (\mathrm{D}_f^{c_i})$$

where $t_1, \ldots, t_k$ are L-terms and in every $f(x_j, \cdot)$, the $\cdot$ is replaced by the right number of L-terms $t(x_1, \ldots, x_{m_i}, \bar{z})$.

For our result, we will also consider relations as well as mutual recursion:

Definition 2.4 Let $f_i : D \times \sigma_1 \times \cdots \times \sigma_l \to \tilde{\sigma_i}$ be new function symbols and $R_i : D \times \sigma_1 \times \cdots \times \sigma_l$ new relation symbols for $i = 1, \ldots n$.

For $A_1, \ldots, A_{n_i} \subseteq \{1, \ldots, n\}$ set

$$\varphi(A_1, \ldots, A_{n_i}) \equiv \bigwedge_{j=1}^{n_i} \Big(\bigwedge_{s \in A_j} R_s(x_j, \bar{z}) \wedge \bigwedge_{s \in A_j^c} \neg R_s(x_j, \bar{z}) \Big)$$

Then a *primitive recursive definition* of the family $\{f_1, \ldots, f_n, R_1, \ldots, R_n\}$ consists of a family of L-terms $\{t_{i,r,\bar{A}} : i \leq k, r \leq n, A_1, \ldots, A_{n_i} \subseteq \{1, \ldots, n\}\}$ and L-formulas $\{\psi_{i,r,\bar{A}} : i \leq k, r \leq n, A_1, \ldots, A_{n_i} \subseteq \{1, \ldots, n\}\}$ with

$$f_r(c_i(x_1, \ldots, x_{m_i}), \bar{z}) = t_{i,r,A}(x_1, \ldots, x_{m_i}, f_1(x_1, \cdot), \ldots f_l(x_{n_i}, \cdot), \bar{z}) \quad (\mathrm{D}_{f_r}^{c_i}(\bar{A}))$$

$$R_r(c_i(x_1, \ldots, x_{m_i}), \bar{z}) \leftrightarrow \psi_{i,r,A}(x_1, \ldots, x_{m_i}, f_1(x_1, \cdot), \ldots f_l(x_{n_i}, \cdot), \bar{z}) \quad (\mathrm{D}_{R_r}^{c_i}(\bar{A}))$$

where all $\cdot$ are replaced by the right number of L-terms $t(x_1, \ldots, x_{m_i}, \bar{z})$.
We also define

$$D_{f_r}^{c_i} \equiv \bigwedge_{A_1, \ldots, A_{n_i} \subseteq \{1 \ldots l\}} \Big(\varphi(\bar{A}) \to D_{f_r}^{c_i}(\bar{A}) \Big)$$

$$D_{R_r}^{c_i} \equiv \bigwedge_{A_1, \ldots, A_{n_i} \subseteq \{1 \ldots l\}} \Big(\varphi(\bar{A}) \to D_{R_r}^{c_i}(\bar{A}) \Big)$$

Like in the natural numbers, for every primitively recursively defined family $\{f_1, \ldots, f_l, R_1, \ldots, R_l\}$, there exist unique interpretations $f_1^{\mathcal{S}}, \ldots, f_l^{\mathcal{S}}, R_1^{\mathcal{S}}, \ldots, R_l^{\mathcal{S}}$ on the standard model $\mathcal{S}$.

The following recursively defined function will play an important role in our result:

Definition 2.5 Let D be an inductive data type with constructors $c_1, \ldots, c_k$. The *length function* $|\cdot| : D \to \mathbb{N}$ is given by

$$|c_i(x_1, \ldots x_{m_i})| = \sum_{j=1}^{n_i} |x_j| + 1 \qquad (\mathrm{D}_{|\cdot|}^{c_i})$$

In order to generalize the notion of straightforward induction, we need induction axioms for inductive data types:

Definition 2.6 Let D be an inductive data type with constuctors $c_1, \ldots, c_k$, $L \supseteq \{c_1, \ldots, c_k\}$ a language and $\varphi(x)$ an L-formula. We define:

$$I_x^{c_i}\varphi(x) \equiv \forall x_1 \cdots \forall x_{m_i} : \Big(\bigwedge_{j=1}^{n_i} \varphi(x_j) \Big) \to \varphi(c_i(x_1, \ldots, x_{m_i}))$$

The *induction axiom* is defined as $I_x^D\varphi(x) \equiv \Big(\bigwedge_{i=1}^k I_x^{c_i}\varphi(x) \Big) \to \forall x : \varphi(x)$.

Again, like in the natural numbers, for every L-formula and any $\mathcal{M}$ that is an expansion of $\mathcal{S}$ to L, $\mathcal{M} \vDash I_x^D \varphi(x)$.

Definition 2.7 Let D be an inductive data type with constructors $c_1, \ldots, c_k$, $L \supseteq \{c_1, \ldots, c_k\}$ a language, T an L-theory and $\varphi(x)$ an L-formula. $\forall x \varphi(x)$ has a *straightforward induction proof* in T iff

$$T, I_x^D \varphi(x) \vdash \forall x \varphi(x)$$

For our result, we also need the notion of the arguments of a function or relation in recursion position that appear in a formula.

Definition 2.8 Let L be a language, $f \in L$. We define recursively:

$$\mathrm{Arg}_f(c) = \mathrm{Arg}_f(x) = \emptyset$$

$$\mathrm{Arg}_f(g(t_1 \ldots t_k)) = \mathrm{Arg}_f(t_1) \cup \cdots \cup \mathrm{Arg}_f(t_k)$$

$$\mathrm{Arg}_f(f(t_1 \ldots t_k)) = \mathrm{Arg}_f(t_1) \cup \cdots \cup \mathrm{Arg}_f(t_k) \cup \{t_1\}$$

$$\mathrm{Arg}_f(R(t_1 \ldots t_k)) = \mathrm{Arg}_f(t_1) \cup \cdots \cup \mathrm{Arg}_f(t_k)$$

$$\mathrm{Arg}_f(\neg\varphi) = \mathrm{Arg}_f(\forall x \varphi) = \mathrm{Arg}_f(\exists x \varphi) = \mathrm{Arg}_f(\varphi)$$

$$\mathrm{Arg}_f(\varphi \wedge \psi) = \mathrm{Arg}_f(\varphi \vee \psi) = \mathrm{Arg}_f(\varphi) \cup \mathrm{Arg}_f(\psi)$$

For a set of functions A we define $\mathrm{Arg}_A(\varphi) = \bigcup_{f \in A} \mathrm{Arg}_f(\varphi)$.

Definition 2.9 Let L be a language, $R \in L$. We define recursively:

$$\mathrm{Arg}_R(S(t_1 \ldots t_k)) = \emptyset$$

$$\mathrm{Arg}_R(R(t_1 \ldots t_k)) = \{t_1\}$$

$$\mathrm{Arg}_R(\neg\varphi) = \mathrm{Arg}_R(\forall x \varphi) = \mathrm{Arg}_R(\exists x \varphi) = \mathrm{Arg}_R(\varphi)$$

$$\mathrm{Arg}_R(\varphi \wedge \psi) = \mathrm{Arg}_R(\varphi \vee \psi) = \mathrm{Arg}_R(\varphi) \cup \mathrm{Arg}_R(\psi)$$

For a set of relations A we define $\mathrm{Arg}_A(\varphi) = \bigcup_{R \in A} \mathrm{Arg}_R(\varphi)$.

3 The main result

Definition 3.1 We call an $L \cup \{f_1, \ldots, f_l, R_1, \ldots, R_l\}$-formula φ *suitable* iff x is the only free variable in φ, x is bound nowhere in the formula and x is the only variable in t for all $t \in \mathrm{Arg}_{\{f_1 \ldots f_l\}}(\varphi) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(\varphi)$.

Definition 3.2 Let D be an inductive data type defined by constructors $c_1, \ldots, c_k$ with $n_1, \ldots, n_k \in \{0, 1\}$, $L \supseteq \{c_1, \ldots, c_k\}$ a language, $\mathcal{S}'$ an expansion of $\mathcal{S}$ to L. Let $f_1, \ldots, f_l, R_1, \ldots, R_l$ be a primitive recursively defined family of new function and relation symbols, $\varphi(x)$ a suitable $L \cup \{f_1, \ldots, f_l, R_1, \ldots, R_l\}$-formula. φ satisfies the *Lundstedt criterion* for lower bound b and counterexample step

constructor c_j, iff for every $m \in \mathbb{N}$ there exist $\alpha_1, \ldots, \alpha_{m_j} \in \mathcal{S}$, $\beta_1, \ldots, \beta_l$ and $\eta_1, \ldots, \eta_l$ interpretations of the new function and relation symbols, such that with $\hat{\mathcal{S}}_m = (\mathcal{S}', \bar{f} \mapsto \bar{\beta}, \bar{R} \mapsto \bar{\eta})$:
1. For all $r \leq l$, $i \leq k$ indices of step constructors and all $\delta_1, \ldots, \delta_{m_i}$ with $|\delta_1| > b : \hat{\mathcal{S}}_m \vDash \forall \bar{z} D^{c_i}_{f_r}(\bar{\delta}, \bar{z}) \wedge \forall \bar{z} D^{c_i}_{R_r}(\bar{\delta}, \bar{z})$
2. $\hat{\mathcal{S}}_m \vDash \varphi(\alpha_1) \wedge \neg\varphi(c_j(\bar{\alpha}))$
3. For all $t \in \mathrm{Arg}_{\{f_1 \ldots f_l\}}(\varphi) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(\varphi)$: $|t^{\hat{\mathcal{S}}_m}(\alpha_1)|, |t^{\hat{\mathcal{S}}_m}(c_j(\bar{\alpha}))| \geq m$

Theorem 3.3 Let D be an inductive data type defined by constructors $c_1, \ldots, c_k$ with $n_1, \ldots, n_k \in \{0, 1\}$, $L \supseteq \{c_1, \ldots, c_k\}$ a language, $\mathcal{S}'$ an expansion of $\mathcal{S}$ to L. Let $f_1, \ldots, f_l, R_1, \ldots, R_l$ be a primitive recursively defined family of new function and relation symbols, $\varphi(x)$ a suitable $L \cup \{f_1, \ldots, f_l, R_1, \ldots, R_l\}$-formula. If there exist a lower bound $b \in \mathbb{N}$ and a step constructor c_j, such that φ satisfies the Lundstedt criterion for b and c_j, then $\forall x \varphi(x)$ does not have a straightforward induction proof in $\mathrm{Th}_L(\mathcal{S}') \cup \{D^{c_i}_{f_p}, D^{c_i}_{R_p} : i \leq k, p \leq l\}$.

For the examples provided by Lundstedt, the following simplified formulation for functions on the natural numbers will suffice:

Corollary 3.4 Let $L \supseteq \{0, s\}$ be a language, with every symbol of L having a fixed interpretation in $\mathbb{N}$. Let $f_1, \ldots, f_l : \mathbb{N}^k \to \mathbb{N}$ be new function symbols,

$$f_i(0, \bar{z}) = t^0_i(\bar{z}) \tag{$\mathrm{D}^0_{f_i}$}$$

$$f_i(s(x), \bar{z}) = t^s_i(x, f_1(x), \ldots, f_l(x), \bar{z}) \tag{$\mathrm{D}^s_{f_i}$}$$

with $t^0_1, \ldots, t^0_l, t^s_1, \ldots, t^s_l$ L-terms and $\varphi(x)$ a suitable $L \cup \{f_1, \ldots, f_l\}$-formula. Suppose for every $m \in \mathbb{N}$ there exists an $\alpha \in \mathbb{N}$ and $\beta_1, \ldots, \beta_l : \mathbb{N}^k \to \mathbb{N}$, such that:
1. $\beta_i(n+1, \bar{z}) = (t^s_i)^{\mathbb{N}}(n, \beta_1(n, \cdot), \ldots, \beta_l(n, \cdot), \bar{z})$ for all $n, z_2, \ldots, z_k \in \mathbb{N}$
2. $\mathbb{N}, \bar{f} \mapsto \bar{\beta} \vDash \varphi(\alpha) \wedge \neg\varphi(\alpha + 1)$
3. $t^{\mathbb{N}, \bar{f} \mapsto \bar{\beta}}(\alpha), t^{\mathbb{N}, \bar{f} \mapsto \bar{\beta}}(\alpha + 1) \geq m$ for all $t \in \mathrm{Arg}_{\{f_1 \ldots f_l\}}(\varphi)$
Then $\forall x \varphi(x)$ does not have a straightforward induction proof in $\mathrm{Th}_L(\mathbb{N}) \cup \{D^0_{f_1}, \ldots, D^0_{f_l}, D^s_{f_1}, \ldots, D^s_{f_l}\}$.

In order to prove the theorem, we first need two lemmas.

Lemma 3.5 Let D be an inductive data type defined by constructors $c_1, \ldots, c_k$ with $n_1, \ldots, n_k \in \{0, 1\}$. Then for every $n \in \mathbb{N}$, there exists a $\{c_1, \ldots, c_k\}$-formula $\psi_n(x)$, such that for all $\alpha \in \mathcal{S} : \mathcal{S} \vDash \psi_n(\alpha)$ iff $|\alpha| = n$.
$\forall \bar{x} : \neg\psi_1(c_i(\bar{x}))$ and $\forall \bar{x} : \neg\psi_n(x_1) \to \neg\psi_{n+1}(c_i(\bar{x}))$ are elements of $\mathrm{Th}_{\{c_1 \ldots c_k\}}(\mathcal{S})$ for all $i \leq k$ indices of step constructors and all $n \in \mathbb{N}$.

Proof. We define ψ_n recursively:

$$\psi_1(x) \equiv \bigvee_{\substack{i \leq k \text{ index of} \\ \text{base constuctor}}} \exists \bar{y} : x = c_i(\bar{y})$$

$$\psi_{n+1} \equiv \bigvee_{\substack{i\leq k \text{ index of} \\ \text{step constuctor}}} \exists \bar{y} : (x = c_i(\bar{y}) \wedge \psi_n(y_1))$$

By induction on n, we directly get that $\mathcal{S} \vDash \psi_n(\alpha)$ implies $|\alpha| = n$. The other direction follows, because every element in $\mathcal{S}$ is reachable by a constructor.

For every step constructor c_i and all $\bar{\alpha} \in \mathcal{S}$ $|c_i(\bar{\alpha})| = |\alpha_1| + 1 \geq 2$, so $\mathcal{S} \vDash \neg\psi_1(c_i(\bar{\alpha}))$.

If $|c_i(\bar{\alpha})| = n + 1$, then we would get $|\alpha_1| = n$, so $\mathcal{S} \vDash \neg\psi_n(\alpha_1) \rightarrow \neg\psi_{n+1}(c_i(\bar{\alpha}))$.

□

Lemma 3.6 Let L be a language, $a_1, \ldots, a_k \in L$ constant symbols, $f_1, \ldots, f_l$, $R_1, \ldots, R_l$ function and relation symbols and $\varphi(\bar{x})$ an $L \cup \{f_1, \ldots, f_l, R_1, \ldots, R_l\}$-formula.
Let $\mathcal{M}$ and $\mathcal{N}$ be models in $L \cup \{f_1, \ldots, f_l, R_1, \ldots, R_l\}$ with the same domain M and matching interpretations $c^{\mathcal{M}} = c^{\mathcal{N}}, f^{\mathcal{M}} = f^{\mathcal{N}}, R^{\mathcal{M}} = R^{\mathcal{N}}$ for all $c, f, R \in L$. Let $A \subseteq M$ be such that $f_1^{\mathcal{M}}\big|_A = f_1^{\mathcal{N}}\big|_A, \ldots, f_l^{\mathcal{M}}\big|_A = f_l^{\mathcal{N}}\big|_A$, $R_1^{\mathcal{M}}\big|_A = R_1^{\mathcal{N}}\big|_A, \ldots, R_l^{\mathcal{M}}\big|_A = R_l^{\mathcal{N}}\big|_A$.
Suppose for every $t \in \mathrm{Arg}_{\{f_1 \ldots f_l\}}(\varphi) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(\varphi) : t^{\mathcal{M}}(\bar{a}^{\mathcal{M}}) \in A$. Then $\mathcal{M} \vDash \varphi(\bar{a})$ iff $\mathcal{N} \vDash \varphi(\bar{a})$.

Proof. We proceed by induction on the subterms and subformulas of φ. For terms t_1, t_2 and formulas ψ_1, ψ_2, define:

$$t_1 \leq_1' t_2 \Leftrightarrow \exists f \exists u_1 \cdots \exists u_k : t_2 = f(u_1, \ldots, u_k) \wedge (t_1 = u_1 \vee \cdots \vee t_1 = u_k)$$

$$\psi_1 \leq_2' \psi_2 \Leftrightarrow \begin{array}{r} (\psi_2 = \neg\psi_1 \vee \psi_2 = \forall x : \psi_1 \vee \psi_2 = \exists x : \psi_1) \vee \\ \exists \psi_3 : ((\psi_2 = \psi_1 \wedge \psi_3) \vee (\psi_2 = \psi_3 \wedge \psi_1)) \vee \\ \exists \psi_3 : ((\psi_2 = \psi_1 \vee \psi_3) \vee (\psi_2 = \psi_3 \vee \psi_1)) \vee \\ \exists \psi_3 : ((\psi_2 = \psi_1 \rightarrow \psi_3) \vee (\psi_2 = \psi_3 \rightarrow \psi_1)) \end{array}$$

$\leq_1, \leq_2$ are the transitive closures of $\leq_1', \leq_2'$ and for a term t and a formula ψ.

$$t \leq_3 \psi \Leftrightarrow \exists R \exists u_1 \cdots \exists u_k : (R(u_1, \ldots, u_k) \leq_2 \psi) \wedge (t \leq_1 u_1 \vee \cdots \vee t \leq_1 u_k)$$

Claim 1: For all $t \leq_3 \varphi(\bar{a}) : t^{\mathcal{M}} = t^{\mathcal{N}}$.

For $t = c \in L$ we demanded $c^{\mathcal{M}} = c^{\mathcal{N}}$. For $t = f(u_1, \ldots, u_k)$, we get $u_1, \ldots, u_k \leq_1 t \leq_3 \varphi(\bar{a})$, so $u_1, \ldots, u_k \leq_3 \varphi(\bar{a})$. By the induction hypothesis, $u_1^{\mathcal{M}} = u_1^{\mathcal{N}}, \ldots, u_k^{\mathcal{M}} = u_k^{\mathcal{N}}$. If $f \in L$, then $f^{\mathcal{M}} = f^{\mathcal{N}}$, so $t^{\mathcal{M}} = t^{\mathcal{N}}$. If $f = f_r$ for $r \leq l$, we know $u_1 \in \mathrm{Arg}_f(t) \subseteq \mathrm{Arg}_f(\varphi(\bar{a}))$ because $t \leq_3 \varphi(\bar{a})$. Therefore $u_1^{\mathcal{M}} \in A$, so $t^{\mathcal{M}} = f^{\mathcal{M}}(u_1, \ldots, u_k) = f^{\mathcal{N}}(u_1, \ldots, u_k) = t^{\mathcal{N}}$. □
Claim 2: For all $\psi \leq_2 \varphi(\bar{a})$ and all $d_1, \ldots, d_k : \mathcal{M} \vDash \psi(\bar{d})$ iff $\mathcal{N} \vDash \psi(\bar{d})$.

For $\psi = R(t_1, \ldots, t_k)$ $\psi \leq_2 \varphi(\bar{a})$ implies $t_1 \ldots, t_k \leq_3 \varphi(\bar{a})$, so $t_1^{\mathcal{M}} = t_1^{\mathcal{N}}, \ldots, t_k^{\mathcal{M}} = t_k^{\mathcal{N}}$. If $R \in L$, then $R^{\mathcal{M}} = R^{\mathcal{N}}$, so for all $d_1, \ldots, d_k : \mathcal{M} \vDash \psi(\bar{d})$ iff $\mathcal{N} \vDash \psi(\bar{d})$. If $R = R_r$ for $r \leq l$ it follows again, that $t_1 \in \mathrm{Arg}_R(\varphi(\bar{a}))$, so

again for all $d_1, \ldots, d_k : \mathcal{M} \vDash \psi(\bar{d})$ iff $\mathcal{N} \vDash \psi(\bar{d})$. For $\psi = \neg\psi_1, \forall x : \psi_1, \exists x : \psi_1, \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \psi_1 \to \psi_2$, the induction step is a standard result. $\square$

Because $\varphi(\bar{a}) \leq_2 \varphi(\bar{a})$ and $\varphi(\bar{a})$ has no free variables, we get $\mathcal{M} \vDash \varphi(\bar{a})$ iff $\mathcal{N} \vDash \varphi(\bar{a})$.

$\square$

Proof of Theorem 3.3 With the formulas $\{\psi_n : n \in \mathbb{N}\}$ we can reformulate points 1-3 to state $\hat{\mathcal{S}}_m \vDash \Gamma \cup \Delta_m$ with $\Gamma = \mathrm{Th}_L(\mathcal{S}) \cup \{\varphi(a_1), \neg\varphi(c_j(a_1, \ldots, a_{m_j}))\} \cup \{\forall \bar{x} : \left(\bigwedge_{p=1}^{b} \neg\psi_p(x_1)\right) \to \left(D_{f_r}^{c_i}(\bar{x}) \wedge D_{R_r}^{c_i}(\bar{x})\right) : r \leq l, i \text{ index of step constructor}\}$ and $\Delta_m = \{\neg\psi_p(t(a_1)) \wedge \neg\psi_q(t(c_j(a_1, \ldots, a_{m_j}))) : p < m, t \in \mathrm{Arg}_{\{f_1 \ldots f_l\}}(\varphi) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(\varphi)\}$.

Then for $\Delta_\infty = \bigcup_{m \in \mathbb{N}} \Delta_m$ and $\Gamma_0 \subseteq \Gamma \cup \Delta_\infty$ finite, there exists an m with $\Gamma_0 \subseteq \Gamma \cup \Delta_m$ and therefore $\hat{\mathcal{S}}_m \vDash \Gamma_0$. So, by the compactness theorem, there exists a model $\mathcal{M}$ with $\mathcal{M} \vDash \Gamma \cup \Delta_\infty$.

Let $A = \{\delta \in M : \delta \text{ has sort } D, \text{ for all } p \in \mathbb{N} : \mathcal{M} \vDash \neg\psi_p(\delta)\}$, then for all $t \in \mathrm{Arg}_{\{f_1 \ldots f_l\}}(\varphi) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(\varphi)$: $t^{\mathcal{M}}(a_1^{\mathcal{M}}), t^{\mathcal{M}}(c_j^{\mathcal{M}}(a_1^{\mathcal{M}}, \ldots, a_{m_j}^{\mathcal{M}}) \in A$ because $\mathcal{M} \vDash \Delta_\infty$.

We define the model $\mathcal{N}$ by $\mathcal{N}|_{L \cup \{a_1, \ldots, a_{m_j}\}} = \mathcal{M}|_{L \cup \{a_1, \ldots, a_{m_j}\}}$. For $r \leq l$, we notice that for all $\delta_1, \bar{\zeta} \in \mathcal{M}$ with $\mathcal{M} \vDash \psi_1(\delta_1)$, we get a unique value for $f_r(\delta_1, \bar{\zeta})$ from one of the $D_{f_r}^{c_i}$. Using all these values, we get a unique value for $f_r(\delta_2, \bar{\zeta})$ with $\mathcal{M} \vDash \psi_2(\delta_2)$ and so on, ending up with a recursively defined f_r^{std} and analogously with R_r^{std}. We define:

$$f_r^{\mathcal{N}}(\delta, \bar{\zeta}) = \begin{cases} f_r^{\mathcal{M}}(\delta, \bar{\zeta}) \text{ if } \delta \in A \\ f_r^{\mathrm{std}}(\delta, \bar{\zeta}) \text{ else} \end{cases}$$

$$R_r^{\mathcal{N}}(\delta, \bar{\zeta}) \equiv \begin{cases} R_r^{\mathcal{M}}(\delta, \bar{\zeta}) \text{ if } \delta \in A \\ R_r^{\mathrm{std}}(\delta, \bar{\zeta}) \text{ else} \end{cases}$$

By Lemma 3.6 $\mathcal{M} \vDash \varphi(a_1)$ iff $\mathcal{N} \vDash \varphi(a_1)$ and $\mathcal{M} \vDash \varphi(c_j(\bar{a}))$ iff $\mathcal{N} \vDash \varphi(c_j(\bar{a}))$. So $\mathcal{N} \vDash \neg I_x^{c_j}\varphi(x)$ and therefore $\mathcal{N} \vDash I_x^D\varphi(x), \neg\forall x : \varphi(x)$.

We also get $\mathcal{N} \vDash \mathrm{Th}_L(\mathcal{S})$ since $\mathcal{N}|_L = \mathcal{M}|_L$.

Finally, we show for all $i \leq k$ and $r \leq l$ $\mathcal{N} \vDash D_{f_r}^{c_i}, D_{R_r}^{c_i}$: Let $\delta_1, \ldots, \delta_{m_i} \in \mathcal{N}$.

If $\forall p \in \mathbb{N} : \mathcal{N} \vDash \neg\psi_p(\delta_1)$, then by Lemma 3.5 $\forall p \in \mathbb{N} : \mathcal{N} \vDash \neg\psi_p(c_i(\bar{\delta}))$, because $\mathcal{N} \vDash \mathrm{Th}_{\{c_1 \ldots c_k\}}(\mathcal{S})$, so $\delta_1, c_i(\bar{\delta}) \in A$. $\mathrm{Arg}_{\{f_1 \ldots f_l\}}(D_{f_r}^{c_i}) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(D_{f_r}^{c_i}) = \mathrm{Arg}_{\{f_1 \ldots f_l\}}(D_{R_r}^{c_i}) \cup \mathrm{Arg}_{\{R_1 \ldots R_l\}}(D_{R_r}^{c_i}) = \{x_1, c_i(\bar{x})\}$, so by Lemma 3.6 $\mathcal{M} \vDash D_{f_r}^{c_i}(\bar{\delta})$ iff $\mathcal{N} \vDash D_{f_r}^{c_i}(\bar{\delta})$ and $\mathcal{M} \vDash D_{R_r}^{c_i}(\bar{\delta})$ iff $\mathcal{N} \vDash D_{R_r}^{c_i}(\bar{\delta})$.

Else, either δ_1 is not of type D, in this case c_i is a base constructor and $\mathcal{M} \vDash \psi_1(c_i(\bar{\delta}))$, or there is a $p \in \mathbb{N}$ with $\mathcal{M} \vDash \psi_p(\delta_1)$, in this case $\mathcal{M} \vDash \psi_{p+1}(c_i(\bar{\delta}))$. In both cases, δ_1 and $c_i(\bar{\delta})$ fall in the recursively defined part of $f^{\mathcal{N}}$ and $R^{\mathcal{N}}$, so they fulfill $D_{f_r}^{c_i}, D_{R_r}^{c_i}$.

In conclusion, we have $\mathcal{N} \vDash \mathrm{Th}_L(\mathcal{S}) \cup \{D_{f_p}^{c_i}, D_{R_p}^{c_i} : i \leq k, p \leq l\}, \mathcal{N} \vDash I_x^D\varphi(x)$ and $\mathcal{N} \nvDash \forall x : \varphi(x)$. So $\mathrm{Th}_L(\mathcal{S}) \cup \{D_{f_p}^{c_i}, D_{R_p}^{c_i} : i \leq k, p \leq l\}, I_x^D\varphi(x) \nvdash \forall x : \varphi(x)$

and therefore $\forall x : \varphi(x)$ does not have a straightforward induction proof in $\mathrm{Th}_L(\mathcal{S}) \cup \{D^{c_i}_{f_p}, D^{c_i}_{R_p} : i \leq k, p \leq l\}$.

□

4 Examples by Lundstedt

We will now apply Corollary 3.4 to the three examples, which Lundstedt examined in his paper. For these examples, the background language does not play a big role in the proofs, one could even consider a language, that includes function and relation symbols for every $f : \mathbb{N}^k \to \mathbb{N}$ and $R \subseteq \mathbb{N}^k$. We will work in the language $L = \{0, s, +, \cdot, <, \exp, !\}$ with $\exp(n, m) = n^m$ and $n! = n(n-1) \cdots 2 \cdot 1$.

Example 4.1 $\forall n \exists k : \sum_{i=0}^{n-1}(2i + 1) = k^2$ does not have a straightforward induction proof.

To see this, we define a new function $f : \mathbb{N} \to \mathbb{N}$ by

$$f(0) = 0 \tag{D^0_f}$$

$$f(s(x)) = f(x) + 2x + 1 \tag{D^s_f}$$

and $\varphi(x) \equiv \exists k : f(x) = k \cdot k$. So, $\mathrm{Arg}_f(\varphi) = \{x\}$. (A more formally correct but less readable version of D^s_f would be $f(s(x)) = (f(x) + (s(s(0)) \cdot x)) + s(0)$, this translation is left to the reader from now on.)

For $m \in \mathbb{N}$ take $\alpha = m$ and $\beta : n \mapsto n^2 + 2m + 1$. Then
1. $\beta(n + 1) = (n + 1)^2 + 2m + 1 = n^2 + 2m + 1 + 2n + 1 = \beta(n) + 2n + 1$.
2. $\beta(m) = m^2 + 2m + 1 = (m + 1)^2$, so $\mathbb{N}, f \mapsto \beta \vDash \varphi(\alpha)$.
But $(m+1)^2 = m^2+2m+1 < \beta(m+1) = m^2+4m+2 < m^2+4m+4 = (m+2)^2$, so there is no k with $\beta(m + 1) = k^2$, so $\mathbb{N}, f \mapsto \beta \vDash \neg\varphi(s(\alpha))$.
3. Finally, $\alpha, s(\alpha) \geq m$ holds by definition.

So, by Corollary 3.4 $\forall x \exists k : f(x) = k \cdot k$ does not have a straightforward induction proof in $\mathrm{Th}_L(\mathbb{N}) \cup \{D^0_f, D^s_f\}$. However, the stronger statement $\forall n : \sum_{i=0}^{n-1}(2i + 1) = n^2$ does have a proof by straightforward induction.

Example 4.2 $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$ is a famous result by Euler and it is easy to check $\frac{\pi^2}{6} < 2$. However, $\forall n : \sum_{i=1}^{n} \frac{1}{i^2} < 2$ does not have a straightforward induction proof.

In order to avoid talking about rational numbers, we will work with the equivalent statement $\forall n : (n!)^2 \sum_{i=1}^{n} \frac{1}{i^2} < 2(n!)^2$. The right and left side of this inequality can be recursively defined as follows:

$$f(0) = 0 \tag{D^0_f}$$

$$g(0) = s(0) \tag{D^0_g}$$

$$f(s(x)) = (x + 1)^2 f(x) + g(x) \tag{D^s_f}$$

$$g(s(x)) = (x+1)^2 g(x) \qquad (\mathrm{D}_g^s)$$

To obtain our original statement, we set $\varphi(x) \equiv f(x) < 2 \cdot g(x)$. Again, $\mathrm{Arg}_f(\varphi) = \mathrm{Arg}_g(\varphi) = \{x\}$.

For $m \in \mathbb{N}$ set $\alpha = m$ and $q = 2 - \frac{1}{2}(\sum_{i=1}^{m} \frac{1}{i^2} + \sum_{i=1}^{m+1} \frac{1}{i^2})$. By simple arithmetic and the result of Euler, it follows that $q + \sum_{i=1}^{m} \frac{1}{i^2} < 2 < \sum_{i=1}^{m+1} \frac{1}{i^2}$ and that $q > 0$, so $q = \frac{q_1}{q_2}$ with $q_1, q_2 \in \mathbb{N}$. We want $\frac{\beta(n)}{\gamma(n)} = q + \sum_{i=1}^{n} \frac{1}{i^2}$, which is achieved by $\beta(n) = (n!)^2 \left(q_2 \sum_{i=1}^{n} \frac{1}{i^2} + q_1\right)$ and $\gamma(n) = (n!)^2 q_2$.
1. $\beta(n+1) = (n+1)^2(n!)^2 \left(q_2 \sum_{i=1}^{n} \frac{1}{i^2} + q_1\right) + (n+1)^2(n!)^2 q_2 \frac{1}{(n+1)^2} = (n+1)^2\beta(n) + \gamma(n)$ and $\gamma(n+1) = (n+1)^2(n!)^2 q_2 = (n+1)^2\gamma(n)$.
2. $(\mathbb{N}, f \mapsto \beta, g \mapsto \gamma \vDash \varphi(n))$
$\Leftrightarrow (n!)^2 \left(q_2 \sum_{i=1}^{n} \frac{1}{i^2} + q_1\right) < 2(n!)^2 q_2$
$\Leftrightarrow \sum_{i=1}^{n} \frac{1}{i^2} + \frac{q_1}{q_2} < 2,$
so $\mathbb{N}, f \mapsto \beta, g \mapsto \gamma \vDash \varphi(\alpha) \wedge \neg\varphi(s(\alpha)$
3. Again, $\alpha, s(\alpha) > m$ is clear.

So, by Corollary 3.4 $\forall x : f(x) < 2g(x)$ does not have a straightforward induction proof in $\mathrm{Th}_L(\mathbb{N}) \cup \{D_f^0, D_g^0, D_f^s, D_g^s\}$. Again, there is a strengthening with a proof by straightforward induction, namely $\forall n : \sum_{i=1}^{n} \frac{1}{i^2} + \frac{1}{n} \leq 2$.

Example 4.3 For many common functions, there are so-called tail-recursive definitions, which involve an accumulator argument a. They may be preferred to the standard recursive definitions for practical reasons. Here are the tail-recursive definitions of the factorial, multiplication and exponentiation:

$$\mathrm{fact}(0, a) = a \qquad (\mathrm{D}_{\mathrm{fact}}^0)$$

$$\mathrm{fact}(s(x), a) = \mathrm{fact}(x, a \cdot s(x)) \qquad (\mathrm{D}_{\mathrm{fact}}^s)$$

$$\mathrm{mult}(x, 0, a) = a \qquad (\mathrm{D}_{\mathrm{mult}}^0)$$

$$\mathrm{mult}(x, s(y), a) = \mathrm{mult}(x, y, a + x) \qquad (\mathrm{D}_{\mathrm{mult}}^s)$$

$$\mathrm{texp}(x, 0, a) = a \qquad (\mathrm{D}_{\mathrm{texp}}^0)$$

$$\mathrm{texp}(x, s(y), a) = \mathrm{texp}(x, y, a \cdot x) \qquad (\mathrm{D}_{\mathrm{texp}}^s)$$

Correctness proofs for these definitions are desirable but not possible using straightforward induction. More precisely, the statements $\forall x : \mathrm{fact(x,1)=x!}$, $\forall x \forall y : \mathrm{mult}(x, y, 0) = x \cdot y$ and $\forall x \forall y : \mathrm{texp}(x, y, 1) = \exp(x, y)$ do not have straightforward induction proofs.

Again, the statements $\forall x \forall a : \mathrm{fact}(x, a) = x! \cdot a$, $\forall y \forall x \forall a : \mathrm{mult}(x, y, a) = x \cdot y + a$ and $\forall y \forall x \forall a : \mathrm{texp}(x, y, a) = \exp(x, y) \cdot a$ admit proofs by straightforward induction and imply the correctness of the definitions.

To apply our criterion, we take $\alpha = m$ and define:

$$\beta(x, a) = m!$$

$$\gamma(x, y, a) = xm$$

$$\gamma'(x,y,a) = my$$

$$\gamma'(x,y,a) = \begin{cases} (m+1)\cdot y + 1 \text{ if } x\cdot y + a = (m+1)\cdot y \\ x\cdot y + a \text{ else} \end{cases}$$

$$\delta(x,y,a) = x^m$$

$$\delta'(x,y,a) = \begin{cases} (m+1)^y + 1 \text{ if } x^y \cdot a = (m+1)^y \\ x^y \cdot a \text{ else} \end{cases}$$

We will only go through the three points of the criterion for the case of mult, fact and exp follow analogously.
1. Because γ only depends on x, $\gamma(x, s(y), a) = \gamma(x, y, a + x)$.
2. $\gamma(x, m, 0) = xm$, but $\gamma(x, s(m), 0) = xm \neq x(m + 1)$ for $x = 1$.
3. $\mathrm{Arg}_{\mathrm{mult}}\big(\forall x : \mathrm{mult}(x, y, 0) = x \cdot y\big) = \{y\}$ and $\alpha, s(\alpha) \geq m$.

This technically only tells us, that there is no proof of $\forall y \forall x : \mathrm{mult}(x, y, 0) = x \cdot y$ by straightforward induction on y. Surprisingly, it is harder to show the nonexistence of a proof by straightforward induction on x, even though induction y seems more natural:
1. $x(y+1)+a = (m+1)y$ iff $xy+(a+x) = (m+1)y$, so either $\gamma'(x, y+1, a) = (m+1)y+1 = \gamma'(x, y, a+x)$ or $\gamma'(x, y+1, a) = x(y+1)+a = \gamma'(x, y, a+x)$.
2. $\gamma'(m, y, 0) = my$, but $\gamma'(s(m), y, 0) = (m+1)y+1 \neq (m+1)y$.
3. $\mathrm{Arg}_{\mathrm{mult}}\big(\forall y : \mathrm{mult}(x, y, 0) = x \cdot y\big) = \{x\}$ and $\alpha, s(\alpha) \geq m$.

5 Further examples

Most times when applying Theorem 3.3 or Corollary 3.4, the simplest approach will be to think of only one function or relation symbol as recursively defined and consider all other symbols as elements of the background language. The following example as well as example 3.4 are counterexamples, where it seems necessary to use the recursive definition of multiple symbols:

Example 5.1 We work in the inductive data type `Nat` of natural numbers and define predicates P and Q by:

$$P(0) \leftrightarrow \top \tag{D^0_P}$$

$$Q(0) \leftrightarrow \top \tag{D^0_Q}$$

$$P(s(x)) \leftrightarrow Q(x) \tag{D^s_P}$$

$$Q(s(x)) \leftrightarrow P(x) \tag{D^s_Q}$$

The above definition is not technically correct, a correct version would look like:

$$P(0) \leftrightarrow \top \tag{D^0_P}$$

$$Q(0) \leftrightarrow \top \tag{D^0_Q}$$

$$\begin{aligned}(P(x) \wedge Q(x) &\to (P(s(x)) \leftrightarrow \top)) \\ \wedge (P(x) \wedge \neg Q(x) &\to (P(s(x)) \leftrightarrow \bot)) \\ \wedge (\neg P(x) \wedge Q(x) &\to (P(s(x)) \leftrightarrow \top)) \\ \wedge (\neg P(x) \wedge \neg Q(x) &\to (P(s(x)) \leftrightarrow \bot))\end{aligned} \tag{D^s_P}$$

$$\begin{aligned}(P(x) \wedge Q(x) &\to (Q(s(x)) \leftrightarrow \top)) \\ \wedge (P(x) \wedge \neg Q(x) &\to (Q(s(x)) \leftrightarrow \top)) \\ \wedge (\neg P(x) \wedge Q(x) &\to (Q(s(x)) \leftrightarrow \bot)) \\ \wedge (\neg P(x) \wedge \neg Q(x) &\to (Q(s(x)) \leftrightarrow \bot))\end{aligned} \tag{D^s_Q}$$

Again, we will from now on only give the more readable version of the definition.

$\forall x : P(x)$ can be proved by induction on $\forall x : P(x) \wedge Q(x)$, but there is no proof by straightforward induction, as can be seen by taking $b = 0$, $\alpha = 2m$ and

$$\eta_1(n) \Leftrightarrow n \text{ is even}$$

$$\eta_2(n) \Leftrightarrow n \text{ is odd}$$

1. Since s is the only step constructor, we have to consider D^s_P and D^s_Q, clearly $s(n)$ is even $\Leftrightarrow$ n is odd and $s(n)$ is odd $\Leftrightarrow$ n is even.
2. $\alpha = 2m$ is even, but $s(\alpha) = 2m + 1$ is not.
3. $\alpha = 2m \geq m$ and $s(\alpha) = 2m + 1 \geq m$.

We will now be working with the inductive data type $\mathtt{List}(s)$, which has the two constructors $\mathrm{nil} : \mathtt{List}(s)^0 \to \mathtt{List}(s)$ and $\mathrm{cons} : \mathtt{List}(s) \times s \to \mathtt{List}(s)$, with nil representing the empty list and $\mathrm{cons}(a, L)$ representing the list with first element a and rest L. We will only consider $\mathtt{List}(\mathtt{Nat})$, but the arguments can be applied to most sorts s. The language L will depend on the example, $\mathcal{S}'$ will be the appropriate expansion of $\mathcal{S}$. We will also use the shorthand $[x_1, \ldots, x_n] = \mathrm{cons}(x_1, \mathrm{cons}(\cdots \mathrm{cons}(x_n, \mathrm{nil}) \cdots))$.

Example 5.2 We can recursively define the concatenation of two lists, as well as the reverse of a list:

$$\mathrm{nil} \circ L_2 = L_2 \tag{$\mathrm{D}^{\mathrm{nil}}_\circ$}$$

$$\mathrm{cons}(x, L_1) \circ L_2 = \mathrm{cons}(x, L_1 \circ L_2) \tag{$\mathrm{D}^{\mathrm{cons}}_\circ$}$$

$$\mathrm{rev}(\mathrm{nil}) = \mathrm{nil} \tag{$\mathrm{D}^{\mathrm{nil}}_{\mathrm{rev}}$}$$

$$\mathrm{rev}(\mathrm{cons}(x, L)) = \mathrm{rev}(L) \circ [x] \tag{$\mathrm{D}^{\mathrm{cons}}_{\mathrm{rev}}$}$$

This is not a primitive recursive definition of the family $\{\circ, \mathrm{rev}\}$ in the sense of Definition 2.4, since $D^{\mathrm{cons}}_{\mathrm{rev}}$ applies $\circ$ to $\mathrm{rev}(L)$ instead of L. So in order to use Theorem 3.3, we need to add $\circ$ to the background language $L = \{\mathrm{nil}, \mathrm{cons}, \circ\}$.

The first two natural statements about rev are $\forall L : \mathrm{rev}(\mathrm{rev}(L)) = L$ and $\forall L_1 \forall L_2 : \mathrm{rev}(L_1 \circ L_2) = \mathrm{rev}(L_2) \circ \mathrm{rev}(L_1)$. The second statement has a proof

by straightforward induction on L_1:

Induction Base: $\mathrm{rev}(\mathrm{nil} \circ L_2) \overset{D_\circ^{\mathrm{nil}}}{=} \mathrm{rev}(L_2) = \mathrm{rev}(L_2) \circ \mathrm{nil} \overset{D_{\mathrm{rev}}^{\mathrm{nil}}}{=} \mathrm{rev}(L_2) \circ \mathrm{rev}(\mathrm{nil})$ $(\forall L : L = L \circ \mathrm{nil} \in \mathrm{Th}_L(\mathcal{S}'))$.

Induction Step: $\mathrm{rev}(\mathrm{cons}(x, L_1) \circ L_2) \overset{D_\circ^{\mathrm{cons}}}{=} \mathrm{rev}(\mathrm{cons}(x, L_1 \circ L_2)) \overset{D_{\mathrm{rev}}^{\mathrm{cons}}}{=} \mathrm{rev}(L_1 \circ L_2) \circ [x] \overset{\mathrm{IH}}{=} (\mathrm{rev}(L_2) \circ \mathrm{rev}(L_1)) \circ [x] = \mathrm{rev}(L_2) \circ (\mathrm{rev}(L_1) \circ [x]) \overset{D_{\mathrm{rev}}^{\mathrm{cons}}}{=} \mathrm{rev}(L_2) \circ \mathrm{rev}(\mathrm{cons}(x, L_1))$ $(\forall L_1 \forall L_2 \forall L_3 : (L_1 \circ L_2) \circ L_3 = L_1 \circ (L_2 \circ L_3) \in \mathrm{Th}_L(\mathcal{S}'))$.

Using the second statement, we also get a proof of the first:

Induction Base: $\mathrm{rev}(\mathrm{rev}(\mathrm{nil})) \overset{D_{\mathrm{rev}}^{\mathrm{nil}}}{=} \mathrm{rev}(\mathrm{nil}) \overset{D_{\mathrm{rev}}^{\mathrm{nil}}}{=} \mathrm{nil}$.

Induction Step: $\mathrm{rev}(\mathrm{rev}(\mathrm{cons}(x, L))) \overset{D_{\mathrm{rev}}^{\mathrm{cons}}}{=} \mathrm{rev}(\mathrm{rev}(L) \circ [x]) = \mathrm{rev}([x]) \circ \mathrm{rev}(\mathrm{rev}(L)) \overset{\mathrm{IH}, D_{\mathrm{rev}}^{\mathrm{cons}}}{=} (\mathrm{nil} \circ [x]) \circ L \overset{D_\circ^{\mathrm{nil}}, D_\circ^{\mathrm{cons}}}{=} \mathrm{cons}(x, L)$.

But the first statement does not have a straightforward induction proof in $\mathrm{Th}_L(\mathcal{S}')$, as we can see by taking $b = 1$ and

$$\beta([x_1, \ldots, x_n]) = [0, x_{n-1}, \ldots, x_1]$$

For $m \in \mathbb{N}$, we take $\alpha_1 = [0, \ldots 0]$ with $|\alpha_1| = m$ and $\alpha_2 = 1$.
1. Since cons is the only step constructor and rev is the only function not in the background language, we only have to consider $D_{\mathrm{rev}}^{\mathrm{cons}}$: $|\delta_1| > 1$ implies $\delta_1 \neq \mathrm{nil}$, so δ_1 is of the form $[k_1, \ldots, k_n]$ with $n > 0$. So $\beta(\mathrm{cons}(\delta_2, \delta_1)) = \beta([\delta_2, k_1, \ldots, k_n]) = [0, k_{n-1}, \ldots, k_1, \delta_2] = \beta(\delta_1) \circ [\delta_2]$.
2. $\beta(\beta(\alpha_1)) = \beta(\beta([0, \ldots, 0])) = \beta([0, \ldots, 0]) = [0, \ldots, 0] = \alpha_1$, but
$\beta(\beta(\mathrm{cons}(\alpha_2, \alpha_1))) = \beta(\beta([1, 0, \ldots, 0])) = \beta([0, \ldots, 0, 1]) = [0, 0, \ldots, 0]$
$\neq \mathrm{cons}(\alpha_2, \alpha_1)$.
3. $\varphi(L) \equiv \mathrm{rev}(\mathrm{rev}(L)) = L$ is an example of a formula with nontrivial arguments: $\mathrm{Arg}_{\mathrm{rev}}(\varphi) = \mathrm{Arg}_{\mathrm{rev}}(\mathrm{rev}(\mathrm{rev}(L))) \cup \mathrm{Arg}_{\mathrm{rev}}(L) = \mathrm{Arg}_{\mathrm{rev}}(\mathrm{rev}(L)) \cup \{\mathrm{rev}(L)\} \cup \emptyset = \{L, \mathrm{rev}(L)\}$.
$|\beta([0, \ldots, 0])| = |[0, \ldots, 0]| = m$ and $|\beta([1, 0, \ldots, 0])| = |[1, 0, \ldots, 0]| = m + 1$.

Example 5.3 There also exists a tail-recursive definition of rev:

$$\mathrm{trev}(\mathrm{nil}, L_2) = L_2 \qquad (\mathrm{D}_{\mathrm{trev}}^{\mathrm{nil}})$$

$$\mathrm{trev}(\mathrm{cons}(x, L_1), L_2) = \mathrm{trev}(L_1, \mathrm{cons}(x, L_2)) \qquad (\mathrm{D}_{\mathrm{trev}}^{\mathrm{cons}})$$

Like in Example 4.3, there is no correctness proof by straightforward induction in $\mathrm{Th}_L(\mathcal{S}') \cup \{D_{\mathrm{trev}}^{\mathrm{nil}}, D_{\mathrm{trev}}^{\mathrm{cons}}\}$ with $L = \{\mathrm{nil}, \mathrm{cons}, \circ, \mathrm{rev}\}$:

Let $\varphi(L) \equiv \mathrm{trev}(L, \mathrm{nil}) = \mathrm{rev}(L), b = 0, m \in \mathbb{N}, \alpha_1 = [0, \ldots, 0]$ with $|\alpha_1| = m, \alpha_2 = 1$ and $\beta(L_1, L_2) = \alpha_1$. Then:
1. Since β is constant, $\beta(\mathrm{cons}(\delta_1, \delta_2), \delta_3) = \beta(\delta_2, \mathrm{cons}(\delta_1, \delta_3))$ for all $\delta_1, \delta_2, \delta_2$.
2. $\beta(\alpha_1, \mathrm{nil}) = \alpha_1 = \mathrm{rev}(\alpha_1)$, but $\beta(\mathrm{cons}(1, \alpha_1), \mathrm{nil}) = \alpha_1 \neq \mathrm{rev}(\alpha_1) \circ [1]$.
3. $|\alpha_1| = m, |\mathrm{cons}(1, \alpha_1)| = m + 1$.

The same b, α_1, α_2 and β also show, that $\forall L : \psi(L)$ does not have a proof by straightforward induction, where $\psi(L) \equiv \mathrm{trev}(\mathrm{trev}(L, \mathrm{nil}), \mathrm{nil}) = L$. It seems that for most tail-recursive definitions, this approach using a constant function

works.

Example 5.4 Theorem 3.3 also makes the analysis of more complex constructions, like the formalization and correctness proof for insertion sort, possible. For this, we will be working in the language $L = \{0, s, \leq, \mathrm{cd}, \mathrm{nil}, \mathrm{cons}\}$, where $\leq$ is the usual order on the natural numbers and cd is a function needed in a case distinction in the definition of the insert function and defined by $\mathrm{cd}(m,n,p,q) = \begin{cases} m \text{ if } p \leq q \\ n \text{ else} \end{cases}$.

We can now define the two functions for insertion of an element into a sorted list and for sorting a list, as well as a relation to indicate that a number is less than or equal to all elements of a list and a predicate for all sorted lists:

$$\mathrm{insert}(x, \mathrm{nil}) = \mathrm{nil} \tag{$\mathrm{D}^{\mathrm{nil}}_{\mathrm{insert}}$}$$

$$\mathrm{insert}(x, \mathrm{cons}(y, L)) = \begin{cases} \mathrm{cons}(x, \mathrm{cons}(y, L)) \text{ if } x \leq y \\ \mathrm{cons}(y, \mathrm{insert}(x, L)) \text{ else} \end{cases} \tag{$\mathrm{D}^{\mathrm{cons}}_{\mathrm{insert}}$}$$

$$\mathrm{isort}(\mathrm{nil}) = \mathrm{nil} \tag{$\mathrm{D}^{\mathrm{nil}}_{\mathrm{isort}}$}$$

$$\mathrm{isort}(\mathrm{cons}(x, L)) = \mathrm{insert}(x, L) \tag{$\mathrm{D}^{\mathrm{cons}}_{\mathrm{isort}}$}$$

$$x \leq' \mathrm{nil} \leftrightarrow \top \tag{$\mathrm{D}^{\mathrm{nil}}_{\leq'}$}$$

$$x \leq' \mathrm{cons}(y, L) \leftrightarrow x \leq y \wedge x \leq' L \tag{$\mathrm{D}^{\mathrm{cons}}_{\leq'}$}$$

$$\mathrm{sorted}(\mathrm{nil}) \leftrightarrow \top \tag{$\mathrm{D}^{\mathrm{nil}}_{\mathrm{sorted}}$}$$

$$\mathrm{sorted}(\mathrm{cons}(x, L)) \leftrightarrow x \leq' L \wedge \mathrm{sorted}(L) \tag{$\mathrm{D}^{\mathrm{cons}}_{\mathrm{sorted}}$}$$

Now we want to show that $\varphi(L) \equiv \mathrm{sorted}(\mathrm{isort}(L))$ does not have a straightforward induction proof. There are two possible ways to do this with slightly different results: one could give variations of the functions or of the predicate.

For the first way, let $b = 1, m \in \mathbb{N}, \alpha_1 = [0, \ldots, 0]$ with $|\alpha_1| = m, \alpha_2 = 1$ and

$$\beta(x, [y_1, \ldots, y_n]) = \mathrm{insert}^{\mathcal{S}'}(x, [y_1, \ldots, y_{n-1}]) \circ [y_n]$$

$$\gamma([x_1, \ldots, x_n]) = \mathrm{isort}^{\mathcal{S}'}([x_1, \ldots, x_{n-1}]) \circ [x_n]$$

1. $|\delta| > 1$ implies $\delta \neq \mathrm{nil}$, so δ is of the form $[k_1, \ldots, k_n]$ with $n > 0$.
If $x \leq y$, then $\beta(x, \mathrm{cons}(y, \delta) = \mathrm{insert}(x, [y, k_1, \ldots, k_{n-1}]) \circ [k_n]$
$= [x, y, k_1, \ldots k_{n-1}] \circ [k_n] = \mathrm{cons}(x, \mathrm{cons}(y, \delta))$, otherwise $\beta(x, \mathrm{cons}(y, \delta)$
$= \mathrm{insert}(x, [y, k_1, \ldots, k_{n-1}]) \circ [k_n] = \mathrm{cons}(y, \mathrm{insert}(x, [k_1, \ldots, k_{n-1}])) \circ [k_n]$
$= \mathrm{cons}(y, \beta(x, \delta))$.
$\gamma(\mathrm{cons}(x, \delta)) = \mathrm{isort}([x, k_1, \ldots, k_{n-1}]) \circ [k_n] = \mathrm{insert}(x, \mathrm{isort}([k_1, \ldots, k_{n-1}])) \circ [k_n] = \beta(x, \mathrm{isort}([k_1, \ldots, k_{n-1}]) \circ [k_n]) = \beta(x, \gamma(\delta))$.
2. $\gamma(\alpha_1) = [0, \ldots, 0]$ is sorted, but $\gamma(\mathrm{cons}(1, \alpha_1)) = [0, \ldots, 0, 1, 0]$ is not.
3. $|\alpha_1|, |\mathrm{cons}(1, \alpha_1)| \geq m$.

So $\varphi(L)$ does not have a straightforward induction proof in $\mathrm{Th}_{L'}(\mathcal{S}')$ with $L' = L \cup \{\leq', \mathrm{sorted}\}$, meaning that the nonexistence of a straightforward induction proof for insertion sort does not depend on the axiomatization of being a sorted list.

For the second way, take $b = 2, m \in \mathbb{N}, \alpha_1 = [0, \ldots, 0]$ with $|\alpha_1| = m, \alpha_2 = 1$ and

$$\eta([x_1, \ldots, x_n]) \Leftrightarrow [x_1, \ldots, x_{n-2}, x_n, x_{n-1}] \text{ is sorted}$$

1. $|\delta| > 2$ implies $\delta = [k_1, \ldots, k_n]$ with $n \geq 2$.
$\eta(\mathrm{cons}(x, \delta)) \Leftrightarrow [x, k_1, \ldots, k_{n-2}, k_n, k_{n-1}]$ is sorted
$\Leftrightarrow x \leq k_1, \ldots, k_n \wedge [k_1, \ldots, k_{n-2}, k_n, k_{n-1}]$ is sorted $\Leftrightarrow x \leq' \delta \wedge \eta(\delta)$
2. $\eta(\mathrm{isort}(\alpha_1)) \Leftrightarrow [0, \ldots, 0]$ is sorted, which is true.
But $\eta(\mathrm{isort}(\mathrm{cons}(1, \alpha_1))) \Leftrightarrow \eta([0, \ldots, 0, 1]) \Leftrightarrow [0, \ldots, 0, 1, 0]$ is sorted, which is false.
3. $|\mathrm{isort}(\alpha_1)| = |[0, \ldots, 0]| = m$ and $|\mathrm{isort}(\mathrm{cons}(1, \alpha_1))| = |[0, \ldots, 0, 1]| = m + 1$.

So $\varphi(L)$ does not have a straightforward induction proof in $\mathrm{Th}_{L'}(\mathcal{S}')$ with $L' = L \cup \{\mathrm{insert}, \mathrm{isort}, \leq'\}$, meaning that the nonexistence of a straightforward induction proof of the correctness of sorted does not depend on the sorting algorithm.

References

[1] Anders Lundstedt. *Necessarily non-analytic induction proofs– summary of some results*. 2020. URL: `https://anderslundstedt.com/research/`.