Logical Foundations of Inductive Theorem Proving

Stefan Hetzl

Institute of Discrete Mathematics and Geometry
TU Wien, Austria

18th International Tbilisi Summer School in Logic and Language

Tbilisi, Georgia

September 12–15, 2024

Introduction

- Induction: mathematics and computer science

Introduction

- Induction: mathematics and computer science
- Automated inductive theorem proving:
  Algorithms for finding proofs by induction

Introduction

- Induction: mathematics and computer science
- Automated inductive theorem proving:
  Algorithms for finding proofs by induction
- Applications in software verification, formal mathematics, ...
- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
- Empirical evaluation of implementations

Introduction

- Induction: mathematics and computer science
- Automated inductive theorem proving:
  Algorithms for finding proofs by induction
- Applications in software verification, formal mathematics, ...
- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
- Empirical evaluation of implementations

- Logical foundations of automated inductive theorem proving
  - E.g., given method M, which theorems can M prove?

Outline

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Proof.

Base case $n = 1$:

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

Step case:

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

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$*.*

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

Step case:

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

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

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

Step case:

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

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

Proof:

$\sum_{i=1}^{n+1} i =$

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

Step case:

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

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

Proof:

$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1)$

A first example: the Gauss sum

Theorem

For all $n \geq 1$: $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

Step case:

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

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

Proof:

$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) =^{\text{IH}} \frac{n(n+1)}{2} + \frac{2(n+1)}{2}$

A first example: the Gauss sum

Theorem

For all $n \geq 1$*:* $\displaystyle\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Proof.

Base case $n = 1$: $1 = \frac{1 \cdot 2}{2}$. ✓

Step case:

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

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

Proof:

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

□

A second example

Theorem

The sum of the first n odd numbers is a square,

A second example

Theorem

The sum of the first n odd numbers is a square, i.e., for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i - 1) = k^2$.

Proof.

Base case $n = 1$:

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ *there is a* $k \in \mathbb{N}$ *s.t.* $\displaystyle\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ *there is a* $k \in \mathbb{N}$ *s.t.* $\displaystyle\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

Proof:

$\sum_{i=1}^{n+1}(2i-1) =$

A second example

Theorem

The sum of the first n odd numbers is a square, i.e., for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

Proof:

$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1)$

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

Proof:

$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1) =^{\mathrm{IH}} k_0^2 + 2n + 1$

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

Proof:

$$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1) =^{\text{IH}} k_0^2 + 2n + 1 = \overset{?}{\cdots} = k_1^2.$$

□

A second example

Theorem

The sum of the first n odd numbers is a square, i.e., for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = k^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

Proof:

$$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1) =^{\mathrm{IH}} k_0^2 + 2n + 1 = \overset{?}{\cdots} = k_1^2. \text{ ✗}$$

We are stuck!

□

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ *there is a* $k \in \mathbb{N}$ *s.t.* $\sum_{i=1}^{n}(2i-1) = n^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\exists k_0 \sum_{i=1}^{n}(2i-1) = k_0^2$

Claim: $\exists k_1 \sum_{i=1}^{n+1}(2i-1) = k_1^2$

Proof:

$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1) =^{\mathrm{IH}} k_0^2 + 2n + 1 = \overset{?}{\cdots} = k_1^2$. ✗

□

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ there is a $k \in \mathbb{N}$ s.t. $\sum_{i=1}^{n}(2i-1) = n^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\sum_{i=1}^{n}(2i-1) = n^2$

Claim: $\sum_{i=1}^{n+1}(2i-1) = (n+1)^2$

Proof:

$$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1) =^{\text{IH}} k_0^2 + 2n + 1 = \overset{?}{\cdots} = k_1^2. \text{ ✗}$$

□

A second example

Theorem

The sum of the first n odd numbers is a square, i.e.,

for all $n \geq 1$ *there is a* $k \in \mathbb{N}$ *s.t.* $\displaystyle\sum_{i=1}^{n}(2i - 1) = n^2$.

Proof.

Base case $n = 1$: $1 = 1^2$. ✓

Step case:

Induction hypothesis: $\sum_{i=1}^{n}(2i-1) = n^2$

Claim: $\sum_{i=1}^{n+1}(2i-1) = (n+1)^2$

Proof:

$\sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + (2n+1) =^{\mathrm{IH}} n^2 + 2n + 1 = (n+1)^2$. ✓

□

Formulating failure (1/2)

Definition

The language of arithmetic is $L_\mathsf{A} = \{0, s, +, \cdot, \leq\}$.

Formulating failure (1/2)

Definition

The language of arithmetic is $L_A = \{0, s, +, \cdot, \leq\}$.

Definition

We define $L_f = L_A \cup \{f/1\}$ and:

$$f(0) = 0 \qquad (D_f^0)$$

$$\forall x\, f(s(x)) = f(x) + (2 \cdot x + 1) \qquad (D_f^+)$$

Formulating failure (1/2)

Definition

The language of arithmetic is $L_{\mathrm{A}} = \{0, s, +, \cdot, \leq\}$.

Definition

We define $L_f = L_{\mathrm{A}} \cup \{f/1\}$ and:

$$f(0) = 0 \qquad (D_f^0)$$

$$\forall x\, f(s(x)) = f(x) + (2 \cdot x + 1) \qquad (D_f^+)$$

Then, in $\mathbb{N}$, $f(n) = \sum_{i=1}^{n} (2i - 1)$.

Formulating failure (2/2)

Definition

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$

Formulating failure (2/2)

Definition

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$

Definition

Let $L \supseteq \{0, s\}$, let $\varphi(x, \overline{z})$ be an L formula, then: $I_x\varphi(x, \overline{z})$ is

$$\forall \overline{z} \big(\varphi(0, \overline{z}) \land \forall x\, (\varphi(x, \overline{z}) \to \varphi(s(x), \overline{z})) \to \forall x\, \varphi(x, \overline{z})\big)$$

Formulating failure (2/2)

Definition

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$

Definition

Let $L \supseteq \{0, s\}$, let $\varphi(x, \overline{z})$ be an L formula, then: $I_x\varphi(x, \overline{z})$ is

$$\forall \overline{z} \big(\varphi(0, \overline{z}) \land \forall x \, (\varphi(x, \overline{z}) \to \varphi(s(x), \overline{z})) \to \forall x \, \varphi(x, \overline{z})\big)$$

Definition

$\psi(x) \equiv \exists y \, f(x) = y \cdot y$

Formulating failure (2/2)

Definition

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$

Definition

Let $L \supseteq \{0, s\}$, let $\varphi(x, \overline{z})$ be an L formula, then: $I_x \varphi(x, \overline{z})$ is

$$\forall \overline{z} \big(\varphi(0, \overline{z}) \land \forall x \, (\varphi(x, \overline{z}) \to \varphi(s(x), \overline{z})) \to \forall x \, \varphi(x, \overline{z})\big)$$

Definition

$\psi(x) \equiv \exists y \, f(x) = y \cdot y$

Theorem (Lundstedt '20)

$T, I_x \psi(x) \nvdash \forall x \, \psi(x)$.

The compactness theorem

Theorem (Compactness theorem)

Let Γ be a set of sentences. If every finite subset of Γ is satisfiable, then Γ is satisfiable.

Theorem (Compactness theorem)

Let Γ be a set of sentences. If every finite subset of Γ is satisfiable, then Γ is satisfiable.

Example

Let $L' = L_{\mathsf{A}} \cup \{c\}$. Define

$$\Gamma = \mathrm{Th}(\mathbb{N}) \cup \{c \geq 0, c \geq 1, c \geq 2, \ldots\}.$$

The compactness theorem

Theorem (Compactness theorem)

Let Γ be a set of sentences. If every finite subset of Γ is satisfiable, then Γ is satisfiable.

Example

Let $L' = L_{\mathrm{A}} \cup \{c\}$. Define

$$\Gamma = \mathrm{Th}(\mathbb{N}) \cup \{c \geq 0, c \geq 1, c \geq 2, \ldots\}.$$

Let $\Gamma_0 \subseteq \Gamma$ be finite. Let $m \in \mathbb{N}$ s.t. $c \geq i \in \Gamma_0$ implies $i < m$.

The compactness theorem

Theorem (Compactness theorem)

Let Γ *be a set of sentences. If every finite subset of* Γ *is satisfiable, then* Γ *is satisfiable.*

Example

Let $L' = L_{\mathsf{A}} \cup \{c\}$. Define

$$\Gamma = \mathrm{Th}(\mathbb{N}) \cup \{c \geq 0, c \geq 1, c \geq 2, \ldots\}.$$

Let $\Gamma_0 \subseteq \Gamma$ be finite. Let $m \in \mathbb{N}$ s.t. $c \geq i \in \Gamma_0$ implies $i < m$.
Define the L' structure $\mathcal{M}_0$ by $\mathcal{M}_0\restriction_{L_{\mathsf{A}}} = \mathbb{N}$ and $c^{\mathcal{M}_0} = m$. Then $\mathcal{M}_0 \models \Gamma_0$.

The compactness theorem

Theorem (Compactness theorem)

Let Γ be a set of sentences. If every finite subset of Γ is satisfiable, then Γ is satisfiable.

Example

Let $L' = L_{\mathsf{A}} \cup \{c\}$. Define

$$\Gamma = \mathrm{Th}(\mathbb{N}) \cup \{c \geq 0, c \geq 1, c \geq 2, \ldots\}.$$

Let $\Gamma_0 \subseteq \Gamma$ be finite. Let $m \in \mathbb{N}$ s.t. $c \geq i \in \Gamma_0$ implies $i < m$.
Define the L' structure $\mathcal{M}_0$ by $\mathcal{M}_0{\restriction_{L_{\mathsf{A}}}} = \mathbb{N}$ and $c^{\mathcal{M}_0} = m$. Then $\mathcal{M}_0 \models \Gamma_0$.
So, by the compactness theorem, there is an $\mathcal{M}$ with $\mathcal{M} \models \Gamma$. Let $\mathcal{N} = \mathcal{M}{\restriction_{L_{\mathsf{A}}}}$, then $\mathcal{N} \models \mathrm{Th}(\mathbb{N})$.

$\mathcal{N}$ is a *nonstandard model* of $\mathrm{Th}(\mathbb{N})$

Standard and Nonstandard numbers

Let $\mathcal{M} \models \mathrm{Th}(\mathbb{N})$.

Standard and Nonstandard numbers

Let $\mathcal{M} \models \mathrm{Th}(\mathbb{N})$.

Definition

Then $m \in \mathcal{M}$ is called *standard* if there is an $n \in \mathbb{N}$ s.t. $s^n(0)^{\mathcal{M}} = m$.
Otherwise m is called non-standard.

Standard and Nonstandard numbers

Let $\mathcal{M} \models \mathrm{Th}(\mathbb{N})$.

Definition

Then $m \in \mathcal{M}$ is called *standard* if there is an $n \in \mathbb{N}$ s.t. $s^n(0)^{\mathcal{M}} = m$. Otherwise m is called non-standard.

Observation

$\mathcal{M} \models \forall x \forall y\, (x \leq y \vee y \leq x)$
$\mathcal{M} \models \forall x\, 0 \leq x$
For all $n \in \mathbb{N}$: $\mathcal{M} \models \forall x\, (x \leq n \rightarrow x = 0 \vee x = 1 \vee \cdots \vee x = n - 1)$

So non-standard m are "after" the standard m.

Proving Failure (1/3)

Definition

Define $L_c = L_f \cup \{c\}$ and

$$\Gamma_c^{+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^{+}, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$$

$$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^{0}, D_f^{+}, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$$

Proving Failure (1/3)

Definition

Define $L_c = L_f \cup \{c\}$ and

$$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$$

$$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$$

Lemma

If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Proving Failure (1/3)

Definition

Define $L_c = L_f \cup \{c\}$ and

$$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$$

$$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$$

Lemma

If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Proof.

For $\mathcal{M} \models \Gamma_c^+$ define $\mathcal{N}$ by $\mathcal{N}\restriction_{L_\mathrm{A} \cup \{c\}} = \mathcal{M}\restriction_{L_\mathrm{A} \cup \{c\}}$ and

$$f^{\mathcal{N}}(x) = \begin{cases} x^2 & \text{if } x \text{ is standard} \\ f^{\mathcal{M}}(x) & \text{otherwise} \end{cases}$$

□

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}.$

Lemma

Γ_c^+ *is satisfiable.*

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \dots\}.$

Lemma

Γ_c^+ is satisfiable.

Definition

For $m \in \mathbb{N}$ define $\beta_m : \mathbb{N} \to \mathbb{N}, n \mapsto n^2 + 2m + 1$.

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.

Lemma

Γ_c^+ *is satisfiable.*

Definition

For $m \in \mathbb{N}$ define $\beta_m : \mathbb{N} \to \mathbb{N}, n \mapsto n^2 + 2m + 1$.

Then $\beta_m(m) = (m+1)^2$ and $\beta_m(m+1)$ is not a square because

$$(m+1)^2 = m^2+2m+1 < \beta_m(m+1) = m^2+4m+2 < m^2+4m+4 = (m+2)^2.$$

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}.$

Lemma

Γ_c^+ *is satisfiable.*

Definition

For $m \in \mathbb{N}$ define $\beta_m : \mathbb{N} \to \mathbb{N}, n \mapsto n^2 + 2m + 1.$

Then $\beta_m(m) = (m+1)^2$ and $\beta_m(m+1)$ is not a square because

$$(m+1)^2 = m^2+2m+1 < \beta_m(m+1) = m^2+4m+2 < m^2+4m+4 = (m+2)^2.$$

Proof.

Let $\Gamma_0 \subseteq \Gamma_c^+$ be finite.

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.

Lemma

Γ_c^+ *is satisfiable.*

Definition

For $m \in \mathbb{N}$ define $\beta_m : \mathbb{N} \to \mathbb{N}, n \mapsto n^2 + 2m + 1$.

Then $\beta_m(m) = (m+1)^2$ and $\beta_m(m+1)$ is not a square because

$$(m+1)^2 = m^2+2m+1 < \beta_m(m+1) = m^2+4m+2 < m^2+4m+4 = (m+2)^2.$$

Proof.

Let $\Gamma_0 \subseteq \Gamma_c^+$ be finite. Let $a \in \mathbb{N}$ s.t. $c \geq i \in \Gamma_0$ implies $i < a$.

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.

Lemma

Γ_c^+ *is satisfiable.*

Definition

For $m \in \mathbb{N}$ define $\beta_m : \mathbb{N} \to \mathbb{N}, n \mapsto n^2 + 2m + 1$.

Then $\beta_m(m) = (m+1)^2$ and $\beta_m(m+1)$ is not a square because

$$(m+1)^2 = m^2+2m+1 < \beta_m(m+1) = m^2+4m+2 < m^2+4m+4 = (m+2)^2.$$

Proof.

Let $\Gamma_0 \subseteq \Gamma_c^+$ be finite. Let $a \in \mathbb{N}$ s.t. $c \geq i \in \Gamma_0$ implies $i < a$. Define the L_c structure $\mathcal{M}_0$ by: $\mathcal{M}_0{\restriction}_{L_\mathrm{A}} = \mathbb{N}$, $c^{\mathcal{M}_0} = a$, $f^{\mathcal{M}_0} = \beta_a$. Then $\mathcal{M}_0 \models \Gamma_0$.

Proving Failure (2/3)

$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.

Lemma

Γ_c^+ *is satisfiable.*

Definition

For $m \in \mathbb{N}$ define $\beta_m : \mathbb{N} \to \mathbb{N}, n \mapsto n^2 + 2m + 1$.

Then $\beta_m(m) = (m+1)^2$ and $\beta_m(m+1)$ is not a square because

$$(m+1)^2 = m^2+2m+1 < \beta_m(m+1) = m^2+4m+2 < m^2+4m+4 = (m+2)^2.$$

Proof.

Let $\Gamma_0 \subseteq \Gamma_c^+$ be finite. Let $a \in \mathbb{N}$ s.t. $c \geq i \in \Gamma_0$ implies $i < a$. Define the L_c structure $\mathcal{M}_0$ by: $\mathcal{M}_0 \restriction_{L_A} = \mathbb{N}$, $c^{\mathcal{M}_0} = a$, $f^{\mathcal{M}_0} = \beta_a$. Then $\mathcal{M}_0 \models \Gamma_0$. So, by compactness, Γ_c^+ is satisfiable. □

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \nvdash \forall x\, \psi(x)$.

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \nvdash \forall x\, \psi(x)$.

Proof.

Let $\mathcal{M} \models \Gamma_c^{0,+}$.

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \not\vdash \forall x\, \psi(x)$.

Proof.

Let $\mathcal{M} \models \Gamma_c^{0,+}$. Let $\mathcal{N} = \mathcal{M}\restriction_{L_f}$.

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \not\vdash \forall x\, \psi(x)$.

Proof.

Let $\mathcal{M} \models \Gamma_c^{0,+}$. Let $\mathcal{N} = \mathcal{M}\restriction_{L_f}$. Then $\mathcal{N} \models T$, $\mathcal{N} \models \psi(0)$,

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \not\vdash \forall x\, \psi(x)$.

Proof.

Let $\mathcal{M} \models \Gamma_c^{0,+}$. Let $\mathcal{N} = \mathcal{M}\restriction_{L_f}$. Then $\mathcal{N} \models T$, $\mathcal{N} \models \psi(0)$, $\mathcal{N} \not\models \forall x\, (\psi(x) \to \psi(s(x)))$ with counterexample $c^{\mathcal{M}}$,

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \not\vdash \forall x\, \psi(x)$.

Proof.

Let $\mathcal{M} \models \Gamma_c^{0,+}$. Let $\mathcal{N} = \mathcal{M}\restriction_{L_f}$. Then $\mathcal{N} \models T$, $\mathcal{N} \models \psi(0)$, $\mathcal{N} \not\models \forall x\,(\psi(x) \rightarrow \psi(s(x))$ with counterexample $c^{\mathcal{M}}$, $\mathcal{N} \not\models \forall x\, \psi(x)$ with counterexample $c^{\mathcal{M}}$.

Proving Failure (3/3)

$T = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+\}$.
$\Gamma_c^+ = \mathrm{Th}(\mathbb{N}) \cup \{D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots\}$.
$\Gamma_c^{0,+} = \mathrm{Th}(\mathbb{N}) \cup \{D_f^0, D_f^+, \psi(c), \neg\psi(s(c)), c \geq 0, c \geq 1, c \geq 2, \ldots$.
Lemma. Γ_c^+ is satisfiable.
Lemma. If Γ_c^+ is satisfiable, then $\Gamma_c^{0,+}$ is satisfiable.

Theorem (Lundstedt '20)

$T, I_x\psi(x) \nvdash \forall x\, \psi(x)$.

Proof.

Let $\mathcal{M} \models \Gamma_c^{0,+}$. Let $\mathcal{N} = \mathcal{M}\restriction_{L_f}$. Then $\mathcal{N} \models T$, $\mathcal{N} \models \psi(0)$, $\mathcal{N} \not\models \forall x\, (\psi(x) \rightarrow \psi(s(x)))$ with counterexample $c^{\mathcal{M}}$, $\mathcal{N} \not\models \forall x\, \psi(x)$ with counterexample $c^{\mathcal{M}}$. So $\mathcal{N} \models I_x\psi(x)$. □

Logical strength

Definition

A formula $\forall x\, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x\, \varphi(x)$.

Logical strength

Definition

A formula $\forall x\, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x\, \varphi(x)$.

- Proof of $\psi(x) \equiv \exists y\, f(x) = y \cdot y$ by induction on $\psi'(x) \equiv f(x) = x \cdot x$.

Logical strength

Definition

A formula $\forall x \, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x \, \varphi(x)$.

- Proof of $\psi(x) \equiv \exists y \, f(x) = y \cdot y$ by induction on $\psi'(x) \equiv f(x) = x \cdot x$.
- Note that $\models \psi'(x) \to \psi(x)$.

Logical strength

Definition

A formula $\forall x\, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x\, \varphi(x)$.

- Proof of $\psi(x) \equiv \exists y\, f(x) = y \cdot y$ by induction on $\psi'(x) \equiv f(x) = x \cdot x$.
- Note that $\models \psi'(x) \to \psi(x)$.
- Do we always / sometimes have to induct on a stronger formula?

Logical strength

Definition

A formula $\forall x\, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x\, \varphi(x)$.

- Proof of $\psi(x) \equiv \exists y\, f(x) = y \cdot y$ by induction on $\psi'(x) \equiv f(x) = x \cdot x$.
- Note that $\models \psi'(x) \to \psi(x)$.
- Do we always / sometimes have to induct on a stronger formula? **No!**

Logical strength

Definition

A formula $\forall x\, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x\, \varphi(x)$.

- Proof of $\psi(x) \equiv \exists y\, f(x) = y \cdot y$ by induction on $\psi'(x) \equiv f(x) = x \cdot x$.
- Note that $\models \psi'(x) \to \psi(x)$.
- Do we always / sometimes have to induct on a stronger formula? **No!**

Observation (H, Wong '18)

T theory. If $T, I_x\varphi(x) \vdash \sigma$ then there is a $\psi(x)$ s.t. $T, I_x\psi(x) \vdash \sigma$ and $T \vdash \forall x\, \psi(x) \leftrightarrow \sigma$.

Logical strength

Definition

A formula $\forall x\, \varphi(x)$ has a *straightforward induction proof* in T if $T, I_x\varphi(x) \vdash \forall x\, \varphi(x)$.

- Proof of $\psi(x) \equiv \exists y\, f(x) = y \cdot y$ by induction on $\psi'(x) \equiv f(x) = x \cdot x$.
- Note that $\models \psi'(x) \to \psi(x)$.
- Do we always / sometimes have to induct on a stronger formula? **No!**

Observation (H, Wong '18)

T theory. If $T, I_x\varphi(x) \vdash \sigma$ then there is a $\psi(x)$ s.t. $T, I_x\psi(x) \vdash \sigma$ and $T \vdash \forall x\, \psi(x) \leftrightarrow \sigma$.

Proof Sketch.

Let $\psi(x) \equiv \varphi(x) \vee \sigma$. □

The number of inductions

Can we strengthen the notion of straightforward induction proof?

Can we prove more with two inductions?

The number of inductions

Can we strengthen the notion of straightforward induction proof?

Can we prove more with two inductions? **No!**

The number of inductions

Can we strengthen the notion of straightforward induction proof?
Can we prove more with two inductions? **No!**

Theorem

T theory. If $T, I_x\varphi_1(x, \overline{z_1}), \dots, I_x\varphi_n(x, \overline{z_n}) \vdash \sigma$, then there is a $\psi(x)$ s.t. $T, I_x\psi(x) \vdash \sigma$.

The number of inductions

Can we strengthen the notion of straightforward induction proof?
Can we prove more with two inductions? **No!**

Theorem

T theory. If $T, I_x\varphi_1(x,\overline{z_1}), \ldots, I_x\varphi_n(x,\overline{z_n}) \vdash \sigma$, then there is a $\psi(x)$ s.t. $T, I_x\psi(x) \vdash \sigma$.

Proof Sketch.

1. Remove parameters by adding universal quantifiers.
2. Pull all inductions together as one. □

The number of inductions

Can we strengthen the notion of straightforward induction proof?
Can we prove more with two inductions? **No!**

Theorem

T theory. If $T, I_x\varphi_1(x,\overline{z_1}), \ldots, I_x\varphi_n(x,\overline{z_n}) \vdash \sigma$, then there is a $\psi(x)$ s.t. $T, I_x\psi(x) \vdash \sigma$.

Proof Sketch.

1. Remove parameters by adding universal quantifiers.
2. Pull all inductions together as one. □

Corollary

T theory. If $T, I_x\varphi_1(x,\overline{z_1}), \ldots, I_x\varphi_n(x,\overline{z_n}) \vdash \sigma$, then there is $\varphi(x)$ s.t. $T, I_x\varphi(x) \vdash \sigma$ and $T \vdash \forall x\, \varphi(x) \leftrightarrow \sigma$.

Outline

Theory exploration

- Automated theorem proving: goal-oriented
  Given T and σ find out if $T \vdash \sigma$

Theory exploration

- Automated theorem proving: goal-oriented
  Given T and σ find out if $T \vdash \sigma$

- Theory exploration: bottom-up
  Given T find "interesting" $\sigma_1, \ldots, \sigma_n$ s.t. $T \vdash \sigma_1$, ..., $T \vdash \sigma_n$

Theory exploration

- Automated theorem proving: goal-oriented
  Given T and σ find out if $T \vdash \sigma$

- Theory exploration: bottom-up
  Given T find "interesting" $\sigma_1, \ldots, \sigma_n$ s.t. $T \vdash \sigma_1$, ..., $T \vdash \sigma_n$

- Equational theory exploration (σ_i are equations)
  - Simplified form of HipSpec
    [Claessen, Johansson, Rosén, Smallbone '13]
  - Allows to "iterate" straightforward induction proofs

Inductive data types

Definition

Work in many-sorted first-order logic with sorts $D, T_1, \ldots, T_n$. D is defined as *inductive data type* by *constructors* $c_1, \ldots, c_k$ where $c_i : \tau_i^1 \times \cdots \times \tau_i^{m_i} \to D$ with $\tau_i^l \in \{D, T_1, \ldots, T_n\}$.

Inductive data types

Definition

Work in many-sorted first-order logic with sorts $D, T_1, \ldots, T_n$. D is defined as *inductive data type* by *constructors* $c_1, \ldots, c_k$ where $c_i : \tau_i^1 \times \cdots \times \tau_i^{m_i} \to D$ with $\tau_i^l \in \{D, T_1, \ldots, T_n\}$.

Example

$D = \mathrm{Nat}$, $n = 0$, $c_1 = 0 : \mathrm{Nat}$, $c_2 = s : \mathrm{Nat} \to \mathrm{Nat}$.

Inductive data types

Definition

Work in many-sorted first-order logic with sorts $D, T_1, \dots, T_n$. D is defined as *inductive data type* by *constructors* $c_1, \dots, c_k$ where $c_i : \tau_i^1 \times \cdots \times \tau_i^{m_i} \to D$ with $\tau_i^l \in \{D, T_1, \dots, T_n\}$.

Example

$D = \mathrm{Nat},\ n = 0,\ c_1 = 0 : \mathrm{Nat},\ c_2 = s : \mathrm{Nat} \to \mathrm{Nat}.$

Example

$D = \mathrm{NatList},\ T_1 = \mathrm{Nat},\ n = 1,\ c_1 = \mathrm{nil} : \mathrm{NatList},$
$c_2 = \mathrm{cons} : \mathrm{Nat} \times \mathrm{NatList} \to \mathrm{NatList}.$

Primitive recursion and induction

Example

Primitive recursion over lists:

$$h(\mathrm{nil}, \overline{z}) = t(\overline{z})$$
$$h(\mathrm{cons}(x, L), \overline{z}) = u(x, L, h(L, \overline{z}), \overline{z})$$

Primitive recursion and induction

Example

Primitive recursion over lists:

$$h(\mathrm{nil}, \overline{z}) = t(\overline{z})$$
$$h(\mathrm{cons}(x, L), \overline{z}) = u(x, L, h(L, \overline{z}), \overline{z})$$

Definition

Let $L = \{c_1, \ldots, c_k\}$. Then a ground L term is called *value*.

Functions defined by primitive recursion evaluate to values.

Primitive recursion and induction

Example

Primitive recursion over lists:

$$h(\mathrm{nil}, \overline{z}) = t(\overline{z})$$
$$h(\mathrm{cons}(x, L), \overline{z}) = u(x, L, h(L, \overline{z}), \overline{z})$$

Definition

Let $L = \{c_1, \ldots, c_k\}$. Then a ground L term is called *value*.

Functions defined by primitive recursion evaluate to values.

Example

The induction axiom for lists: $\varphi(X, \overline{z})$ formula:

$$\forall \overline{z} \big(\varphi(\mathrm{nil}, \overline{z}) \land \forall X\, \forall u\, (\varphi(X, \overline{z}) \to \varphi(\mathrm{cons}(u, X), \overline{z})) \to \forall X\, \varphi(X, \overline{z})\big)$$

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

$$\text{len}(\text{nil}) = 0$$
$$\text{len}(\text{cons}(x, L)) = s(\text{len}(L))$$

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

$$\mathrm{len}(\mathrm{nil}) = 0 \qquad \mathrm{app}(\mathrm{nil}, L_2) = L_2$$
$$\mathrm{len}(\mathrm{cons}(x, L)) = s(\mathrm{len}(L)) \qquad \mathrm{app}(\mathrm{cons}(x, L_1), L2) = \mathrm{cons}(x, \mathrm{app}(L_1, L_2))$$

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

$$\text{len}(\text{nil}) = 0 \qquad \text{app}(\text{nil}, L_2) = L_2$$
$$\text{len}(\text{cons}(x, L)) = s(\text{len}(L)) \quad \text{app}(\text{cons}(x, L_1), L2) = \text{cons}(x, \text{app}(L_1, L_2))$$

- $L_1 : 0 + x = x$ has straightforward (sf) induction proof

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

$$\mathrm{len}(\mathrm{nil}) = 0 \qquad \mathrm{app}(\mathrm{nil}, L_2) = L_2$$
$$\mathrm{len}(\mathrm{cons}(x, L)) = s(\mathrm{len}(L)) \quad \mathrm{app}(\mathrm{cons}(x, L_1), L2) = \mathrm{cons}(x, \mathrm{app}(L_1, L_2))$$

- $L_1 : 0 + x = x$ has straightforward (sf) induction proof
- $L_2 : s(x) + y = s(x + y)$ has sf induction proof

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

$$\mathrm{len}(\mathrm{nil}) = 0 \qquad \mathrm{app}(\mathrm{nil}, L_2) = L_2$$
$$\mathrm{len}(\mathrm{cons}(x, L)) = s(\mathrm{len}(L)) \quad \mathrm{app}(\mathrm{cons}(x, L_1), L2) = \mathrm{cons}(x, \mathrm{app}(L_1, L_2))$$

- $L_1 : 0 + x = x$ has straightforward (sf) induction proof
- $L_2 : s(x) + y = s(x + y)$ has sf induction proof
- $L_3 : x + y = y + x$ has sf induction proof using L_1 and L_2.

Example

- Datatypes: Nat $(0, s)$, NatList (nil, cons)
- Defined functions

$$x + 0 = x$$
$$x + s(y) = s(x + y)$$

$$\mathrm{len}(\mathrm{nil}) = 0 \qquad \mathrm{app}(\mathrm{nil}, L_2) = L_2$$
$$\mathrm{len}(\mathrm{cons}(x, L)) = s(\mathrm{len}(L)) \quad \mathrm{app}(\mathrm{cons}(x, L_1), L2) = \mathrm{cons}(x, \mathrm{app}(L_1, L_2))$$

- $L_1 : 0 + x = x$ has straightforward (sf) induction proof
- $L_2 : s(x) + y = s(x + y)$ has sf induction proof
- $L_3 : x + y = y + x$ has sf induction proof using L_1 and L_2.
- $L_4 : \mathrm{len}(\mathrm{app}(L_1, L_2)) = \mathrm{len}(L_1) + \mathrm{len}(L_2)$ has sf induction proof using L_1 and L_2.

Generating equational conjectures

procedure $\textsc{Conjecture}(k, \overline{x}, n)$

$T := \{t \text{ term} \mid |t| \leq k, \mathrm{Var}(t) \subseteq \{\overline{x}\}\}$

$E := \{(t_1, t_2) \mid t_1, t_2 \in T\}$

return $\{t_1 = t_2 \mid (t_1, t_2) \in E\}$

end procedure

- $t_1 = t_2$ is returned iff $t_1 = t_2$ withstood n tests

Generating equational conjectures

procedure $\textsc{Conjecture}(k, \overline{x}, n)$
  $T := \{t \text{ term} \mid |t| \leq k, \mathrm{Var}(t) \subseteq \{\overline{x}\}\}$
  $E := \{(t_1, t_2) \mid t_1, t_2 \in T\}$
  for $i := 1, \ldots, n$ **do**
    $\overline{a} := \textsc{GenerateRandomTuple}(\overline{x})$
  end for
  return $\{t_1 = t_2 \mid (t_1, t_2) \in E\}$
end procedure

- $t_1 = t_2$ is returned iff $t_1 = t_2$ withstood n tests

Generating equational conjectures

procedure $\textsc{Conjecture}(k, \overline{x}, n)$
  $T := \{t \text{ term} \mid |t| \leq k, \mathrm{Var}(t) \subseteq \{\overline{x}\}\}$
  $E := \{(t_1, t_2) \mid t_1, t_2 \in T\}$
  for $i := 1, \ldots, n$ **do**
    $\overline{a} := \textsc{GenerateRandomTuple}(\overline{x})$
    for each equivalence class C of E **do**
    end for
  end for
  return $\{t_1 = t_2 \mid (t_1, t_2) \in E\}$
end procedure

- $t_1 = t_2$ is returned iff $t_1 = t_2$ withstood n tests

Generating equational conjectures

procedure $\textsc{Conjecture}(k, \overline{x}, n)$
  $T := \{t \text{ term} \mid |t| \leq k, \mathrm{Var}(t) \subseteq \{\overline{x}\}\}$
  $E := \{(t_1, t_2) \mid t_1, t_2 \in T\}$
  for $i := 1, \ldots, n$ **do**
    $\overline{a} := \textsc{GenerateRandomTuple}(\overline{x})$
    for each equivalence class C of E **do**
      $E' := \{(t_1, t_2) \in C \mid \textsc{Value}(t_1[\overline{x}\backslash\overline{a}]) = \textsc{Value}(t_2[\overline{x}\backslash\overline{a}])\}$
      Replace C by E' in E
    end for
  end for
  return $\{t_1 = t_2 \mid (t_1, t_2) \in E\}$
end procedure

- $t_1 = t_2$ is returned iff $t_1 = t_2$ withstood n tests

Equational theory exploration

procedure $\textsc{Explore}(A, k, \overline{x}, n, t)$
  $L := \emptyset$
  $C := \textsc{Conjecture}(k, \overline{x}, n)$

  return L
end procedure

Equational theory exploration

procedure $\textsc{Explore}(A, k, \overline{x}, n, t)$
$L := \emptyset$
$C := \textsc{Conjecture}(k, \overline{x}, n)$
while $C \neq \emptyset$ **do**
Pick $\varphi(x_1, \ldots, x_m) \in C$
$C := C \setminus \{\varphi(\overline{x})\}$

end while
return L
end procedure

Equational theory exploration

procedure $\textsc{Explore}(A, k, \overline{x}, n, t)$

$L := \emptyset$

$C := \textsc{Conjecture}(k, \overline{x}, n)$

while $C \neq \emptyset$ **do**

Pick $\varphi(x_1, \ldots, x_m) \in C$

$C := C \setminus \{\varphi(\overline{x})\}$

if $A, L \nvdash^t \forall \overline{x}\, \varphi(\overline{x})$ **then**

end if

end while

return L

end procedure

Equational theory exploration

procedure $\textsc{Explore}(A, k, \overline{x}, n, t)$
    $L := \emptyset$
    $C := \textsc{Conjecture}(k, \overline{x}, n)$
    while $C \neq \emptyset$ **do**
        Pick $\varphi(x_1, \ldots, x_m) \in C$
        $C := C \setminus \{\varphi(\overline{x})\}$
        if $A, L \not\vdash^t \forall \overline{x}\, \varphi(\overline{x})$ **then**
            if $\exists i \in \{1, \ldots, m\}$ s.t. $A, L, I_{x_i}\varphi(\overline{x}) \vdash^t \forall \overline{x}\, \varphi(\overline{x})$ **then**
                $L := L \cup \{\forall \overline{x}\, \varphi(\overline{x})\}$
            end if
        end if
    end while
    return L
end procedure

Evaluation

- Simple algorithm

Evaluation

- Simple algorithm
- Useful in practice
  inductive data types and simple primitive recursive functions
- Finds commutation properties, simple lemmas, ...

Evaluation

- Simple algorithm
- Useful in practice
  inductive data types and simple primitive recursive functions
- Finds commutation properties, simple lemmas, ...
- Main weakness: limited to equations (atoms)

Outline

1. Straightforward induction proofs
2. Equational theory exploration
3. Atomic induction
4. Literal induction
5. Saturation theorem proving with explicit induction axioms
6. Open induction
7. Clause set cycles
8. Existential induction
9. Conclusion

Atomic induction

Definition

For a set of formulas Γ define

$$\Gamma\text{-IND} = \{ I_x \varphi(x, \bar{z}) \mid \varphi(x, \bar{z}) \in \Gamma \}.$$

Remark

Γ-IND *goes beyond straightforward induction proofs.*

Atomic induction

Definition

For a set of formulas Γ define

$$\Gamma\text{-IND} = \{I_x\varphi(x,\bar{z}) \mid \varphi(x,\bar{z}) \in \Gamma\}.$$

Remark

Γ-IND *goes beyond straightforward induction proofs.*

Example

Atom-IND are all induction axioms with atoms as induction formula.

Observation

Everything provable by equational theory exploration is provable by atomic induction.

What does atomic induction prove?

Example

Let $L_{\mathrm{LA}} = \{0, s, p, +\}$ and $B =$

$$s(x) \neq 0 \qquad x + 0 = x$$
$$p(0) = 0 \qquad x + s(y) = s(x + y)$$
$$p(s(x)) = x$$

What does atomic induction prove?

Example

Let $L_{\mathrm{LA}} = \{0, s, p, +\}$ and $B =$

$$s(x) \neq 0 \qquad x + 0 = x$$
$$p(0) = 0 \qquad x + s(y) = s(x + y)$$
$$p(s(x)) = x$$

Then $B, \text{Atom-IND} \vdash \forall x \forall y\, x + y = y + x$

What does atomic induction prove?

Example

Let $L_{\mathrm{LA}} = \{0, s, p, +\}$ and $B =$

$$s(x) \neq 0 \qquad x + 0 = x$$
$$p(0) = 0 \qquad x + s(y) = s(x + y)$$
$$p(s(x)) = x$$

Then $B, \text{Atom-IND} \vdash \forall x \forall y\, x + y = y + x$

$B, \text{Atom-IND} \vdash \forall x \forall y \forall z\, x + (y + z) = (x + y) + z.$

A nonstandard model of atomic induction (1/2)

Definition

Define the L_{LA}-structure $\mathcal{M}$ with domain $\mathbb{N} \cup \{\infty\}$ by interpreting $0, s, p, +$ on $\mathbb{N}$ in the standard way and

$$s^{\mathcal{M}}(\infty) = \infty = p^{\mathcal{M}}(\infty) \text{ and } n +^{\mathcal{M}} \infty = \infty +^{\mathcal{M}} n = \infty +^{\mathcal{M}} \infty = \infty.$$

A nonstandard model of atomic induction (1/2)

Definition

Define the L_{LA}-structure $\mathcal{M}$ with domain $\mathbb{N} \cup \{\infty\}$ by interpreting $0, s, p, +$ on $\mathbb{N}$ in the standard way and

$$s^{\mathcal{M}}(\infty) = \infty = p^{\mathcal{M}}(\infty) \text{ and } n +^{\mathcal{M}} \infty = \infty +^{\mathcal{M}} n = \infty +^{\mathcal{M}} \infty = \infty.$$

Observation

$\mathcal{M} \models B$.

A nonstandard model of atomic induction (2/2)

Observation

$\mathcal{M} \models$ Atomic-IND.

A nonstandard model of atomic induction (2/2)

Observation

$\mathcal{M} \models$ Atomic-IND.

Proof.

Let $\overline{z} = z_1, \ldots, z_k$, $t_1(x, \overline{z}) = t_2(x, \overline{z})$ atom,

A nonstandard model of atomic induction (2/2)

Observation

$\mathcal{M} \models$ Atomic-IND.

Proof.

Let $\overline{z} = z_1, \ldots, z_k$, $t_1(x, \overline{z}) = t_2(x, \overline{z})$ atom, $\overline{a} \in (\mathbb{N} \cup \{\infty\})^k$. Assume

(I) $\mathcal{M} \models t_1(0, \overline{a}) = t_2(0, \overline{a})$ and

(II) $\mathcal{M} \models \forall x \, (t_1(x, \overline{a}) = t_2(x, \overline{a}) \rightarrow t_1(s(x), \overline{a}) = t_2(s(x), \overline{a}))$

Claim. $\mathcal{M} \models t_1(b, \overline{a}) = t_2(b, \overline{a})$ for all $b \in \mathbb{N} \cup \{\infty\}$.

A nonstandard model of atomic induction (2/2)

Observation

$\mathcal{M} \models$ Atomic-IND.

Proof.

Let $\overline{z} = z_1, \ldots, z_k$, $t_1(x, \overline{z}) = t_2(x, \overline{z})$ atom, $\overline{a} \in (\mathbb{N} \cup \{\infty\})^k$. Assume

(I) $\mathcal{M} \models t_1(0, \overline{a}) = t_2(0, \overline{a})$ and

(II) $\mathcal{M} \models \forall x\, (t_1(x, \overline{a}) = t_2(x, \overline{a}) \to t_1(s(x), \overline{a}) = t_2(s(x), \overline{a}))$

Claim. $\mathcal{M} \models t_1(b, \overline{a}) = t_2(b, \overline{a})$ for all $b \in \mathbb{N} \cup \{\infty\}$.

1. $\infty \in \{a_1, \ldots, a_k, b\}$: $\mathcal{M} \models t_1(b, \overline{a}) = \infty = t_2(b, \overline{a})$.

A nonstandard model of atomic induction (2/2)

Observation

$\mathcal{M} \models$ Atomic-IND.

Proof.

Let $\overline{z} = z_1, \ldots, z_k$, $t_1(x, \overline{z}) = t_2(x, \overline{z})$ atom, $\overline{a} \in (\mathbb{N} \cup \{\infty\})^k$. Assume

(I) $\mathcal{M} \models t_1(0, \overline{a}) = t_2(0, \overline{a})$ and

(II) $\mathcal{M} \models \forall x\, (t_1(x, \overline{a}) = t_2(x, \overline{a}) \rightarrow t_1(s(x), \overline{a}) = t_2(s(x), \overline{a}))$

Claim. $\mathcal{M} \models t_1(b, \overline{a}) = t_2(b, \overline{a})$ for all $b \in \mathbb{N} \cup \{\infty\}$.

1. $\infty \in \{a_1, \ldots, a_k, b\}$: $\mathcal{M} \models t_1(b, \overline{a}) = \infty = t_2(b, \overline{a})$.
2. $a_1, \ldots, a_k, b \in \mathbb{N}$: obtain $\mathcal{M} \models t_1(b, \overline{a}) = t_2(b, \overline{a})$ by (I) and b instances of (II). □

Independence results for atomic induction

Observation

$\mathcal{M} \not\models \forall x\, s(x) \neq x$.

Independence results for atomic induction

Observation

$\mathcal{M} \not\models \forall x\, s(x) \neq x$.

Observation

$\mathcal{M} \not\models \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z)$

Proof.

$\infty + 1 = \infty + 2$ but $1 \neq 2$. □

Independence results for atomic induction

Observation

$\mathcal{M} \not\models \forall x\, s(x) \neq x.$

Observation

$\mathcal{M} \not\models \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z)$

Proof.

$\infty + 1 = \infty + 2$ but $1 \neq 2$. □

Corollary

B + Atomic-IND $\not\vdash \forall x\, s(x) \neq x$ *and*
B + Atomic-IND $\not\vdash \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z)$.

Outline

Literal induction

Definition

A *literal* is an atom or a negated atom.

Definition

Literal-IND is the set of induction axioms for literals als induction formulas.

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \to u = v.$

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Proof.

Work in B: □

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Proof.

Work in B: If $s(u) = s(v)$ then $p(s(u)) = p(s(v))$. □

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Proof.

Work in B: If $s(u) = s(v)$ then $u = p(s(u)) = p(s(v)) = v$. □

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Proof.

Induction on x in $s(x) \neq x$. Work in B:

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Proof.

Induction on x in $s(x) \neq x$. Work in B:

1. $s(0) \neq 0$ because $\forall x\, s(x) \neq 0$

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Proof.

Induction on x in $s(x) \neq x$. Work in B:

1. $s(0) \neq 0$ because $\forall x\, s(x) \neq 0$
2. $s(x) \neq x \rightarrow s(s(x)) \neq s(x)$ because $s(s(x)) = s(x) \rightarrow s(x) = x$. □

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Theorem

$B, \text{Literal-IND} \vdash \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z).$

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Theorem

$B, \text{Literal-IND} \vdash \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z).$

Proof.

Assume $y \neq z$. Induction on x in $x + y \neq x + z$. Work in B:

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \to u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Theorem

$B, \text{Literal-IND} \vdash \forall x \forall y \forall z\, (x + y = x + z \to y = z).$

Proof.

Assume $y \neq z$. Induction on x in $x + y \neq x + z$. Work in B:

1. $y \neq z$.

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Theorem

$B, \text{Literal-IND} \vdash \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z).$

Proof.

Assume $y \neq z$. Induction on x in $x + y \neq x + z$. Work in B:

1. $0 + y = y \neq z = 0 + z.$

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \rightarrow u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Theorem

$B, \text{Literal-IND} \vdash \forall x \forall y \forall z\, (x + y = x + z \rightarrow y = z).$

Proof.

Assume $y \neq z$. Induction on x in $x + y \neq x + z$. Work in B:

1. $0 + y = y \neq z = 0 + z.$
2. If $x + y \neq x + z$, then $s(x + y) \neq s(x + z)$. □

What does literal induction prove?

Lemma

$B \vdash s(u) = s(v) \to u = v.$

Theorem

$B, \text{Literal-IND} \vdash \forall x\, s(x) \neq x.$

Theorem

$B, \text{Literal-IND} \vdash \forall x \forall y \forall z\, (x + y = x + z \to y = z).$

Proof.

Assume $y \neq z$. Induction on x in $x + y \neq x + z$. Work in B:

1. $0 + y = y \neq z = 0 + z.$
2. If $x + y \neq x + z$, then $s(x) + y = s(x + y) \neq s(x + z) = s(x) + z.$ □

Independence result for literal induction

Definition

Let $T_{\mathrm{EO}} = \{\ 0 \neq s(x), s(x) = s(y) \rightarrow x = y,$
$E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x))\ \}.$

Independence result for literal induction

Definition

Let $T_{\mathrm{EO}} = \{\ 0 \neq s(x), s(x) = s(y) \rightarrow x = y,$
$E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x))\ \}.$

Observation

$\forall x\,(E(x) \vee O(x))$ *has straightforward induction proof in* T_{EO}.

Independence result for literal induction

Definition

Let $T_{\mathrm{EO}} = \{\ 0 \neq s(x), s(x) = s(y) \rightarrow x = y,$
$E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x))\ \}.$

Observation

$\forall x\,(E(x) \vee O(x))$ has straightforward induction proof in T_{EO}.

Theorem

$T_{\mathrm{EO}} + \mathit{Literal}\text{-IND} \nvdash \forall x\,(E(x) \vee O(x))$.

Independence result for literal induction

Definition

Let $T_{\mathrm{EO}} = \{\ 0 \neq s(x), s(x) = s(y) \to x = y,$
$E(0), E(x) \to O(s(x)), O(x) \to E(s(x))\ \}$.

Observation

$\forall x\,(E(x) \vee O(x))$ *has straightforward induction proof in* T_{EO}.

Theorem

$T_{\mathrm{EO}} + \mathit{Literal}\text{-IND} \nvdash \forall x\,(E(x) \vee O(x))$.

Proof Sketch.

Model $\mathcal{M}$ with domain $(\{0\} \times \mathbb{N}) \cup (\{1\} \times \mathbb{Z})$ and

$$0^{\mathcal{M}} = (0,0) \qquad E^{\mathcal{M}} = \{(0,n) \mid n \text{ is even}\}$$
$$s^{\mathcal{M}}(b,n) = (b,n+1) \qquad O^{\mathcal{M}} = \{(0,n) \mid n \text{ is odd}\}$$

□

Outline

Clause logic

Standard setting for automated theorem proving in first-order logic.

Clause logic

Standard setting for automated theorem proving in first-order logic.

Definition

A *clause* is a formula $\bigvee_{j=1}^{k} L_j$ where L_j literal. A *conjunctive normal form* is a formula $\forall \overline{x} \bigwedge_{i=1}^{n} \bigvee_{i=j}^{k_i} L_{i,j}$ where $L_{i,j}$ literal.

We identify a clause set with a conjunctive normal form.

Clause logic

Standard setting for automated theorem proving in first-order logic.

Definition

A *clause* is a formula $\bigvee_{j=1}^{k} L_j$ where L_j literal. A *conjunctive normal form* is a formula $\forall \overline{x} \bigwedge_{i=1}^{n} \bigvee_{i=j}^{k_i} L_{i,j}$ where $L_{i,j}$ literal.

We identify a clause set with a conjunctive normal form.

Definition

Clause form transformation: given a FOL formula φ we compute

$$\neg\varphi \quad \mapsto \quad \mathrm{sk}^{\exists}(\neg\varphi) \quad \mapsto \quad \mathrm{CNF}(\mathrm{sk}^{\exists}(\neg\varphi)).$$

Then φ is valid iff $\mathrm{CNF}(\mathrm{sk}(\neg\varphi))$ is unsatisfiable.

Skolemisation

Idea: $\forall x \exists y\, \varphi(x, y) \mapsto \forall x\, \varphi(x, f(x))$ where f new function symbol

Skolemisation

Idea: $\forall x \exists y\, \varphi(x, y) \mapsto \forall x\, \varphi(x, f(x))$ where f new function symbol

Definition

The *Skolem axiom* for $\varphi(\overline{x}, y)$ is $\forall \overline{x}\, (\exists y\, \varphi(\overline{x}, y) \to \varphi(\overline{x}, f(\overline{x})))$.

Skolemisation

Idea: $\forall x \exists y\, \varphi(x, y) \mapsto \forall x\, \varphi(x, f(x))$ where f new function symbol

Definition

The *Skolem axiom* for $\varphi(\overline{x}, y)$ is $\forall \overline{x}\, (\exists y\, \varphi(\overline{x}, y) \rightarrow \varphi(\overline{x}, f(\overline{x})))$.

Definition

Skolem closure of a language L is $\mathrm{sk}^{\omega}(L)$.

Skolemisation

Idea: $\forall x \exists y\, \varphi(x, y) \mapsto \forall x\, \varphi(x, f(x))$ where f new function symbol

Definition

The *Skolem axiom* for $\varphi(\overline{x}, y)$ is $\forall \overline{x}\, (\exists y\, \varphi(\overline{x}, y) \to \varphi(\overline{x}, f(\overline{x})))$.

Definition

Skolem closure of a language L is $\mathrm{sk}^{\omega}(L)$.

Definition

$\mathrm{sk}^{\exists}(\varphi)$ is the formula φ after removel of all (positive) existential (and negative universal) quantifiers by Skolemisation.

Skolemisation

Idea: $\forall x \exists y\, \varphi(x, y) \mapsto \forall x\, \varphi(x, f(x))$ where f new function symbol

Definition

The *Skolem axiom* for $\varphi(\overline{x}, y)$ is $\forall \overline{x}\, (\exists y\, \varphi(\overline{x}, y) \rightarrow \varphi(\overline{x}, f(\overline{x})))$.

Definition

Skolem closure of a language L is $\mathrm{sk}^{\omega}(L)$.

Definition

$\mathrm{sk}^{\exists}(\varphi)$ is the formula φ after removel of all (positive) existential (and negative universal) quantifiers by Skolemisation.

Theorem

$\mathrm{sk}^{\omega}(L)$-SA $\vdash \varphi \leftrightarrow \mathrm{sk}^{\exists}(\varphi)$.

Saturation theorem proving

Standard technique for automated theorem proving in FOL

Saturation theorem proving

Standard technique for automated theorem proving in FOL

Definition

Saturation system $\mathcal{S}$ is a set of rules for deriving new clauses from the current clause set.

Saturation theorem proving

Standard technique for automated theorem proving in FOL

Definition

Saturation system $\mathcal{S}$ is a set of rules for deriving new clauses from the current clause set.

Example

The *resolution rule* is

$$\frac{C \vee L \quad L' \vee D}{(C \vee D)\sigma}$$

where σ is most general unifier of L and $\overline{L'}$.

Saturation theorem proving

Standard technique for automated theorem proving in FOL

Definition

Saturation system $\mathcal{S}$ is a set of rules for deriving new clauses from the current clause set.

Example

The *resolution rule* is

$$\frac{C \vee L \quad L' \vee D}{(C \vee D)\sigma}$$

where σ is most general unifier of L and $\overline{L'}$.

Example

$$\frac{P(a) \quad \neg P(x) \vee P(f(x))}{P(f(a))}$$

Soundness and refutational completeness

Definition

Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all *n*-ary rules $\rho \in \mathcal{S}$:

$$C_1, \ldots, C_n \in \mathcal{C} \text{ implies } \rho(C_1, \ldots, C_n) \in \mathcal{C}$$

Soundness and refutational completeness

Definition

Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all *n*-ary rules $\rho \in \mathcal{S}$:

$$C_1, \ldots, C_n \in \mathcal{C} \text{ implies } \rho(C_1, \ldots, C_n) \in \mathcal{C}$$

Given $\mathcal{C}$, compute closure by $\mathcal{C}^0 = \mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \ldots \longrightarrow \mathcal{C}^\omega$.

Soundness and refutational completeness

Definition

Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all *n*-ary rules $\rho \in \mathcal{S}$:

$$C_1, \ldots, C_n \in \mathcal{C} \text{ implies } \rho(C_1, \ldots, C_n) \in \mathcal{C}$$

Given $\mathcal{C}$, compute closure by $\mathcal{C}^0 = \mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \ldots \longrightarrow \mathcal{C}^\omega$.

Definition

$\mathcal{S}$ *sound* if $C \in \mathcal{C}^\omega$ implies $\mathcal{C} \models C$

Soundness and refutational completeness

Definition

Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all n-ary rules $\rho \in \mathcal{S}$:

$$C_1, \ldots, C_n \in \mathcal{C} \text{ implies } \rho(C_1, \ldots, C_n) \in \mathcal{C}$$

Given $\mathcal{C}$, compute closure by $\mathcal{C}^0 = \mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \ldots \longrightarrow \mathcal{C}^\omega$.

Definition

$\mathcal{S}$ *sound* if $C \in \mathcal{C}^\omega$ implies $\mathcal{C} \models C$

Definition

$\mathcal{S}$ *refutationally complete* if $\mathcal{C} \models \bot$ implies $\bot \in \mathcal{C}^\omega$

Soundness and refutational completeness

Definition

Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all n-ary rules $\rho \in \mathcal{S}$:

$$C_1, \ldots, C_n \in \mathcal{C} \text{ implies } \rho(C_1, \ldots, C_n) \in \mathcal{C}$$

Given $\mathcal{C}$, compute closure by $\mathcal{C}^0 = \mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \ldots \longrightarrow \mathcal{C}^\omega$.

Definition

$\mathcal{S}$ *sound* if $C \in \mathcal{C}^\omega$ implies $\mathcal{C} \models C$

Definition

$\mathcal{S}$ *refutationally complete* if $\mathcal{C} \models \bot$ implies $\bot \in \mathcal{C}^\omega$

Sound and refutationally complete saturation systems for ATP in FOL.

Adding explicit induction axioms

Definition

The general induction rule is

$$\frac{}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \overline{z})))}$$

Adding explicit induction axioms

Definition

The general induction rule is

$$\frac{}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \overline{z})))}$$

Example

$$\forall \overline{z} \big(\varphi(0, \overline{z}) \land \forall x \, (\varphi(x, \overline{z}) \to \varphi(s(x), \overline{z})) \to \forall x \, \varphi(x, \overline{z}) \big)$$

is mapped by $\mathrm{sk}^{\exists}$ to:

$$\forall \overline{z} \big(\mathrm{sk}^{\forall}(\varphi(0, \overline{z})) \land \big(\mathrm{sk}^{\exists}(\varphi(f(\overline{z}), \overline{z})) \to \mathrm{sk}^{\forall}(\varphi(s(f(\overline{z})), \overline{z})) \big) \to \forall x \, \mathrm{sk}^{\exists}(\varphi(x, \overline{z})) \big)$$

Adding explicit induction axioms

Definition

The general induction rule is

$$\frac{}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \overline{z})))}$$

Example

$$\forall \overline{z} \big(\varphi(0, \overline{z}) \land \forall x \, (\varphi(x, \overline{z}) \to \varphi(s(x), \overline{z})) \to \forall x \, \varphi(x, \overline{z}) \big)$$

is mapped by $\mathrm{sk}^{\exists}$ to:

$$\forall \overline{z} \big(\mathrm{sk}^{\forall}(\varphi(0, \overline{z})) \land \big(\mathrm{sk}^{\exists}(\varphi(f(\overline{z}), \overline{z})) \to \mathrm{sk}^{\forall}(\varphi(s(f(\overline{z})), \overline{z})) \big) \to \forall x \, \mathrm{sk}^{\exists}(\varphi(x, \overline{z})) \big)$$

Remark

The general induction rule adds new (Skolem) symbols to the language.

Adding explicit induction axioms

Definition

The general induction rule is

$$\frac{}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \bar{z})))}$$

Example

$$\forall \bar{z} \big(\varphi(0, \bar{z}) \land \forall x \, (\varphi(x, \bar{z}) \to \varphi(s(x), \bar{z})) \to \forall x \, \varphi(x, \bar{z}) \big)$$

is mapped by $\mathrm{sk}^{\exists}$ to:

$$\forall \bar{z} \big(\mathrm{sk}^{\forall}(\varphi(0, \bar{z})) \land \big(\mathrm{sk}^{\exists}(\varphi(f(\bar{z}), \bar{z})) \to \mathrm{sk}^{\forall}(\varphi(s(f(\bar{z})), \bar{z})) \big) \to \forall x \, \mathrm{sk}^{\exists}(\varphi(x, \bar{z})) \big)$$

Remark

The general induction rule adds new (Skolem) symbols to the language. This is iterated.

Adding explicit induction axioms

Definition

The general induction rule is

$$\frac{}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x\varphi(x,\overline{z})))}$$

Example

$$\forall\overline{z}\big(\varphi(0,\overline{z}) \land \forall x\,(\varphi(x,\overline{z}) \to \varphi(s(x),\overline{z})) \to \forall x\,\varphi(x,\overline{z})\big)$$

is mapped by $\mathrm{sk}^{\exists}$ to:

$$\forall\overline{z}\big(\mathrm{sk}^{\forall}(\varphi(0,\overline{z})) \land \big(\mathrm{sk}^{\exists}(\varphi(f(\overline{z}),\overline{z})) \to \mathrm{sk}^{\forall}(\varphi(s(f(\overline{z})),\overline{z}))\big) \to \forall x\,\mathrm{sk}^{\exists}(\varphi(x,\overline{z}))\big)$$

Remark

The general induction rule adds new (Skolem) symbols to the language. This is iterated. Difficult to describe in terms of the original language.

Concrete realisation: Vampire

Definition

Vampire prover [Voronkov et al. '20]: single clause induction

$$\frac{\overline{L(a)} \lor C}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x L(x)))}\ \mathrm{SCIND}$$

a constant symbol, $L(x)$ literal, x only variable in $L(x)$

Concrete realisation: Vampire

Definition

Vampire prover [Voronkov et al. '20]: single clause induction

$$\frac{\overline{L(a)} \vee C}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x L(x)))} \ \mathrm{SCIND}$$

a constant symbol, $L(x)$ literal, x only variable in $L(x)$

Example

$\{x + 0 = 0, x + s(y) = s(x + y), c + (c + c) \neq (c + c) + c\}$ solved by $\mathcal{S} + \mathrm{SCIND}$. Includes generalisation!

Concrete realisation: Vampire

Definition

Vampire prover [Voronkov et al. '20]: single clause induction

$$\frac{\overline{L(a)} \vee C}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x L(x)))} \ \mathrm{SCIND}$$

a constant symbol, $L(x)$ literal, x only variable in $L(x)$

Example

$\{x + 0 = 0, x + s(y) = s(x + y), c + (c + c) \neq (c + c) + c\}$ solved by $\mathcal{S}$ + SCIND. Includes generalisation!

$$I_x L(x) \equiv L(0) \wedge \forall x\,(L(x) \rightarrow L(s(x))) \rightarrow \forall x\, L(x)$$
$$\mathrm{sk}^{\exists}(I_x L(x)) \equiv L(0) \wedge (L(c) \rightarrow L(s(c))) \rightarrow \forall x\, L(x)$$

Concrete realisation: Vampire

Definition

Vampire prover [Voronkov et al. '20]: single clause induction

$$\frac{\overline{L(a)} \vee C}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x L(x)))}\ \mathrm{SCIND}$$

a constant symbol, $L(x)$ literal, x only variable in $L(x)$

Example

$\{x + 0 = 0, x + s(y) = s(x + y), c + (c + c) \neq (c + c) + c\}$ solved by $\mathcal{S} + \mathrm{SCIND}$. Includes generalisation!

$$I_x L(x) \equiv L(0) \wedge \forall x\,(L(x) \rightarrow L(s(x))) \rightarrow \forall x\, L(x)$$
$$\mathrm{sk}^{\exists}(I_x L(x)) \equiv L(0) \wedge (L(c) \rightarrow L(s(c))) \rightarrow \forall x\, L(x)$$

Does not leave "ground induction".

Characterisation (1/2)

Definition

Φ set of formulas. The *ground induction rule* is

$$\frac{C_1 \quad \cdots \quad C_n}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \overline{t})))} \ \Phi\text{-GIND}$$

where $\varphi(x, \overline{z}) \in \Phi$, $\overline{t}$ ground $L(\{C_1, \dots, C_n\})$ terms

Characterisation (1/2)

Definition

Φ set of formulas. The *ground induction rule* is

$$\frac{C_1 \quad \cdots \quad C_n}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x\varphi(x,\overline{t})))}\ \Phi\text{-GIND}$$

where $\varphi(x,\overline{z}) \in \Phi$, $\overline{t}$ ground $L(\{C_1,\ldots,C_n\})$ terms

Lemma

$\mathcal{S}$ sound saturation system, T theory, Φ set of formulas. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$, then $\mathrm{sk}^{\omega}(\mathrm{L}(T) \cup \mathrm{L}(\Phi) \cup \{0, s\})$-SA $+ T + \Phi$-IND is inconsistent.

Characterisation (1/2)

Definition

Φ set of formulas. The *ground induction rule* is

$$\frac{C_1 \quad \cdots \quad C_n}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \bar{t})))} \ \Phi\text{-GIND}$$

where $\varphi(x, \bar{z}) \in \Phi$, $\bar{t}$ ground $L(\{C_1, \ldots, C_n\})$ terms

Lemma

$\mathcal{S}$ sound saturation system, T theory, Φ set of formulas. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$, then $\mathrm{sk}^{\omega}(\mathrm{L}(T) \cup \mathrm{L}(\Phi) \cup \{0, s\})$-SA $+ T + \Phi$-IND is inconsistent.

Proof Sketch.

Translate $\mathcal{S} + \Phi$-GIND refutation line by line. □

Characterisation (1/2)

Definition

Φ set of formulas. The *ground induction rule* is

$$\frac{C_1 \quad \cdots \quad C_n}{\mathrm{CNF}(\mathrm{sk}^{\exists}(I_x \varphi(x, \overline{t})))} \; \Phi\text{-GIND}$$

where $\varphi(x, \overline{z}) \in \Phi$, $\overline{t}$ ground $L(\{C_1, \ldots, C_n\})$ terms

Lemma

$\mathcal{S}$ sound saturation system, T theory, Φ set of formulas. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$, then $\mathrm{sk}^{\omega}(\mathrm{L}(T) \cup \mathrm{L}(\Phi) \cup \{0, s\})$-SA $+ T + \Phi$-IND is inconsistent.

Corollary

$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow \Psi$-IND. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \Psi$-IND is inconsistent.

Characterisation (2/2)

Lemma.
$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow$ Ψ-IND. If $\mathcal{S} + \Phi$-GIND refutes CNF(sk$^{\exists}(T)$) then $T + \Psi$-IND is inconsistent.

Characterisation (2/2)

Lemma.
$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow$ Ψ-IND. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \Psi$-IND is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \text{Literal-IND}$ is inconsistent.

Characterisation (2/2)

Lemma.
$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow$ Ψ-IND. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \Psi$-IND is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \text{Literal-IND}$ is inconsistent.

Proof.

$\mathcal{S} + \mathrm{Literal}(\mathrm{L}(\mathrm{sk}^{\exists}(T)))$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$.

Characterisation (2/2)

Lemma.
$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow$ Ψ-IND. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \Psi$-IND is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \mathrm{Literal\text{-}IND}$ is inconsistent.

Proof.

$\mathcal{S} + \mathrm{Literal}(\mathrm{L}(\mathrm{sk}^{\exists}(T)))$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$.

$\mathrm{L}(\mathrm{sk}^{\exists}(T)) = \mathrm{L}(T) \cup \Sigma$ with Σ constants

Characterisation (2/2)

Lemma.
$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow$ Ψ-IND. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \Psi$-IND is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \text{Literal-IND}$ is inconsistent.

Proof.

$\mathcal{S} + \mathrm{Literal}(\mathrm{L}(\mathrm{sk}^{\exists}(T)))$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$.

$\mathrm{L}(\mathrm{sk}^{\exists}(T)) = \mathrm{L}(T) \cup \Sigma$ with Σ constants , so
$\mathrm{Literal}(\mathrm{L}(T))$-IND $\Leftrightarrow$ $\mathrm{Literal}(\mathrm{L}(\mathrm{sk}^{\exists}(T)))$-IND.

Characterisation (2/2)

Lemma.
$\mathcal{S}$ sound saturation system, T Skolem-free theory, Φ set of formulas, Ψ Skolem-free set of formulas with Φ-IND $\Leftrightarrow$ Ψ-IND. If $\mathcal{S} + \Phi$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \Psi$-IND is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \text{Literal-IND}$ is inconsistent.

Proof.

$\mathcal{S} + \mathrm{Literal}(\mathrm{L}(\mathrm{sk}^{\exists}(T)))$-GIND refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$.

$\mathrm{L}(\mathrm{sk}^{\exists}(T)) = \mathrm{L}(T) \cup \Sigma$ with Σ constants , so
$\mathrm{Literal}(\mathrm{L}(T))$-IND $\Leftrightarrow$ $\mathrm{Literal}(\mathrm{L}(\mathrm{sk}^{\exists}(T)))$-IND.

So, by Lemma, $T + \mathrm{Literal}(\mathrm{L}(T))$-IND is inconsistent. □

Independence result for single clause induction

Theorem. $T_{\mathrm{EO}} + \mathit{Literal}\text{-IND} \nvdash \forall x\,(E(x) \lor O(x))$.

Independence result for single clause induction

Theorem. $T_{\mathrm{EO}} + \mathit{Literal}\text{-IND} \nvdash \forall x\,(E(x) \vee O(x))$.

Theorem. $\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \text{Literal-IND}$ is inconsistent.

Independence result for single clause induction

Theorem. $T_{\mathrm{EO}} + \mathit{Literal}\text{-IND} \nvdash \forall x\,(E(x) \lor O(x))$.

Theorem. $\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \text{Literal-IND}$ is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system. $\mathcal{S} + \mathrm{SCIND}$ does not refute $\mathrm{CNF}(\mathrm{sk}^{\exists}(T_{\mathrm{EO}} + \exists x\,(\neg E(x) \land \neg O(x))))$.

Independence result for single clause induction

Theorem. $T_{\mathrm{EO}} + \mathit{Literal}\text{-IND} \nvdash \forall x\,(E(x) \vee O(x))$.

Theorem. $\mathcal{S}$ sound saturation system, T Skolem-free $\exists_2$ theory. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathrm{CNF}(\mathrm{sk}^{\exists}(T))$ then $T + \mathrm{Literal\text{-}IND}$ is inconsistent.

Theorem

$\mathcal{S}$ sound saturation system. $\mathcal{S} + \mathrm{SCIND}$ does not refute $\mathrm{CNF}(\mathrm{sk}^{\exists}(T_{\mathrm{EO}} + \exists x\,(\neg E(x) \wedge \neg O(x))))$.

Proof.

$T_{\mathrm{EO}} + \exists x\,(\neg E(x) \wedge \neg O(x)) + \mathrm{Literal\text{-}IND}$ is consistent. □

Outline

Open induction

Definition

A formula φ is called *open* if it does not contain quantifiers.

Definition

Open induction is Open-IND.

Open induction

Definition

A formula φ is called *open* if it does not contain quantifiers.

Definition

Open induction is Open-IND.

Theorem (Shoenfield '58)

Over the L_{LA} *theory* $B = \{s(x) \neq 0, p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y)\}$, *open induction (in* L_{LA}*) is equivalent to:*

$$x + y = y + x \qquad x = 0 \lor x = s(p(x))$$

$$(x + y) + z = x + (y + z) \qquad x + y = x + z \to y = z$$

Literal induction vs. open induction

Theorem

$B +$ Literal-IND $\Leftrightarrow B +$ Open-IND.

Literal induction vs. open induction

Theorem

$B + \text{Literal-IND} \Leftrightarrow B + \text{Open-IND}$.

Proof.

Show finite axiomatisation of $B + \text{Open-IND}$ in $B + \text{Literal-IND}$. □

Literal induction vs. open induction

Theorem

$B + \text{Literal-IND} \Leftrightarrow B + \text{Open-IND}$.

Proof.

Show finite axiomatisation of $B + \text{Open-IND}$ in $B + \text{Literal-IND}$. □

Theorem (Weiser '24)

For T natural base theory in $L = \{0, s, p, +, \cdot\}$:

$$T + \text{Literal-IND} \not\Leftrightarrow T + \text{Open-IND}.$$

List cancellation

Sequences with concatenation operation $\frown$

List cancellation

Sequences with concatenation operation $\frown$

Observation

Finite sequences have the properties:

- *left cancellation:* $X \frown Y = X \frown Z \rightarrow Y = Z$
- *right cancellation:* $Y \frown X = Z \frown X \rightarrow Y = Z$

List cancellation

Sequences with concatenation operation $\frown$

Observation

Finite sequences have the properties:

- *left cancellation:* $X \frown Y = X \frown Z \to Y = Z$
- *right cancellation:* $Y \frown X = Z \frown X \to Y = Z$

Observation

Infinite (ω-)sequences satisfy:

- *left cancellation*

but not

- *right cancellation, e.g.* $a^\omega = (a) \frown a^\omega = \mathrm{nil} \frown a^\omega$ *but* $(a) \neq \mathrm{nil}$

A theory of lists with concatenation

Definition

$\mathcal{L}_1 = \{\text{nil} : \text{list}, \text{cons} : \iota \times \text{list} \to \text{list}, \frown : \text{list} \times \text{list} \to \text{list}\}$, $T_1 =$

$$\text{nil} \neq \text{cons}(x, X)$$
$$\text{cons}(x, X) = \text{cons}(y, Y) \to x = y \land X = Y$$
$$\text{nil} \frown Y = Y$$
$$\text{cons}(x, X) \frown Y = \text{cons}(x, X \frown Y)$$

Transfinite sequences

Definition

A sequence of length α is mapping from α to X where α ordinal (in this talk: $\alpha < \omega^3$), X any set.

Definition

Flattening $\lfloor \mathfrak{l} \rfloor$ of a sequence of sequences, e.g.

$$\lfloor((1\ 2\ 3 \cdots)(2\ 3\ 5 \cdots))\rfloor = (1\ 2\ 3 \cdots 2\ 3\ 5 \cdots)$$

Definition

For $a \in X^\alpha$ write a^β for $\lfloor (a)_{\gamma<\beta} \rfloor$, i.e., β times the sequence a.

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \rightarrow Y = Z$

Proof.

It suffices to show that $T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \rightarrow Y = \mathrm{nil}$.

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \rightarrow Y = Z$

Proof.

It suffices to show that $T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \rightarrow Y = \mathrm{nil}$.

Define $N_k = (k, k+1, k+2, \ldots)$ infinite $(\omega-)$sequence,

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \text{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \to Y = Z$

Proof.

It suffices to show that $T_1 + \text{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \to Y = \text{nil}$.

Define $N_k = (k, k+1, k+2, \ldots)$ infinite ($\omega-$)sequence,

$\mathcal{N} = \{w \frown N_k \mid w \in \mathbb{N}^*, k \in \mathbb{N}\}$, decomposition unique, and

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \rightarrow Y = Z$

Proof.

It suffices to show that $T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \rightarrow Y = \mathrm{nil}$.

Define $N_k = (k, k+1, k+2, \ldots)$ infinite ($\omega-$)sequence,
$\mathcal{N} = \{w \frown N_k \mid w \in \mathbb{N}^*, k \in \mathbb{N}\}$, decomposition unique, and
$\mathfrak{L} = \{\lfloor \mathfrak{l} \rfloor \frown w \mid w \in \mathbb{N}^*, \mathfrak{l} \in \mathcal{N}^\beta, \beta < \omega^2\}$.

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \rightarrow Y = Z$

Proof.

It suffices to show that $T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \rightarrow Y = \mathrm{nil}$.

Define $N_k = (k, k+1, k+2, \ldots)$ infinite ($\omega-$)sequence,

$\mathcal{N} = \{w \frown N_k \mid w \in \mathbb{N}^*, k \in \mathbb{N}\}$, decomposition unique, and

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

Define $\mathcal{L}_1$-structure M_2 by $M_2(\mathrm{list}) = \mathfrak{L}$ with nil^{M_2}, cons^{M_2}, $\frown^{M_2}$ having natural interpretation

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \to Y = Z$

Proof.

It suffices to show that $T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \to Y = \mathrm{nil}$.

Define $N_k = (k, k+1, k+2, \ldots)$ infinite ($\omega-$)sequence,
$\mathcal{N} = \{w \frown N_k \mid w \in \mathbb{N}^*, k \in \mathbb{N}\}$, decomposition unique, and
$\mathfrak{L} = \{\lfloor \mathfrak{l} \rfloor \frown w \mid w \in \mathbb{N}^*, \mathfrak{l} \in \mathcal{N}^\beta, \beta < \omega^2\}$.

Define $\mathcal{L}_1$-structure M_2 by $M_2(\mathrm{list}) = \mathfrak{L}$ with nil^{M_2}, cons^{M_2}, $\frown^{M_2}$ having natural interpretation

Then $M_2 \models T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND}$ but
$M_2 \not\models Y \frown X = X \to Y = \mathrm{nil}$

Unprovability of right cancellation

Theorem (H, Vierling '24)

$T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = Z \frown X \to Y = Z$

Proof.

It suffices to show that $T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND} \nvdash Y \frown X = X \to Y = \mathrm{nil}$.

Define $N_k = (k, k+1, k+2, \ldots)$ infinite ($\omega-$)sequence,

$\mathcal{N} = \{w \frown N_k \mid w \in \mathbb{N}^*, k \in \mathbb{N}\}$, decomposition unique, and

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

Define $\mathcal{L}_1$-structure M_2 by $M_2(\mathrm{list}) = \mathfrak{L}$ with nil^{M_2}, cons^{M_2}, $\frown^{M_2}$ having natural interpretation

Then $M_2 \models T_1 + \mathrm{Open}(\mathcal{L}_1)\text{-IND}$ but

$M_2 \not\models Y \frown X = X \to Y = \mathrm{nil}$

Counterexample: $N_0 \in \mathfrak{L}$, $N_0^\omega = \lfloor (N_0)_{\alpha<\omega} \rfloor \in \mathfrak{L}$,

$N_0 \frown N_0^\omega = N_0^\omega$ but $N_0 \neq \mathrm{nil}$. □

Outline

Clause set cycles

Abstraction of *n*-clause calculus [Kersani, Peltier '13; Kersani '14]

Clause set cycles

Abstraction of *n*-clause calculus [Kersani, Peltier '13; Kersani '14]

Definition

An $L \cup \{\eta\}$ clause set $\mathcal{C}$ is a *clause set cycle (CSC)* if $\mathcal{C}(s(\eta)) \models \mathcal{C}(\eta)$ and $\mathcal{C}(0) \models \bot$. An $L \cup \{\eta\}$ clause set $\mathcal{D}(\eta)$ is refuted by a CSC $\mathcal{C}(\eta)$ if $\mathcal{D}(\eta) \models \mathcal{C}(\eta)$.

Clause set cycles

Abstraction of *n*-clause calculus [Kersani, Peltier '13; Kersani '14]

Definition

An $L \cup \{\eta\}$ clause set $\mathcal{C}$ is a *clause set cycle (CSC)* if $\mathcal{C}(s(\eta)) \models \mathcal{C}(\eta)$ and $\mathcal{C}(0) \models \bot$. An $L \cup \{\eta\}$ clause set $\mathcal{D}(\eta)$ is refuted by a CSC $\mathcal{C}(\eta)$ if $\mathcal{D}(\eta) \models \mathcal{C}(\eta)$.

Many equivalent variants.

Clause set cycles

Abstraction of *n*-clause calculus [Kersani, Peltier '13; Kersani '14]

Definition

An $L \cup \{\eta\}$ clause set $\mathcal{C}$ is a *clause set cycle (CSC)* if $\mathcal{C}(s(\eta)) \models \mathcal{C}(\eta)$ and $\mathcal{C}(0) \models \bot$. An $L \cup \{\eta\}$ clause set $\mathcal{D}(\eta)$ is refuted by a CSC $\mathcal{C}(\eta)$ if $\mathcal{D}(\eta) \models \mathcal{C}(\eta)$.

Many equivalent variants.

Example

CSC solves Even/Odd example.

Logical Characterisation

Definition

Γ set of formulas, define

$$\frac{\varphi(0) \quad \varphi(x) \to \varphi(s(x))}{\varphi(\eta)} \; \Gamma\text{-IND}_{\eta}^{\mathrm{R}-}$$

where $\varphi(x) \in \Gamma$.

Logical Characterisation

Definition

Γ set of formulas, define

$$\frac{\varphi(0) \quad \varphi(x) \rightarrow \varphi(s(x))}{\varphi(\eta)} \; \Gamma\text{-IND}_{\eta}^{\mathrm{R}-}$$

where $\varphi(x) \in \Gamma$.

Definition

T theory, R inference rule, define $[T, R] = T + \{\varphi \mid T \vdash \Gamma, \Gamma/\varphi \in R\}$.

Logical Characterisation

Definition

Γ set of formulas, define

$$\frac{\varphi(0) \quad \varphi(x) \rightarrow \varphi(s(x))}{\varphi(\eta)} \ \Gamma\text{-IND}_\eta^{\mathrm{R}-}$$

where $\varphi(x) \in \Gamma$.

Definition

T theory, R inference rule, define $[T, R] = T + \{\varphi \mid T \vdash \Gamma, \Gamma/\varphi \in R\}$.

Theorem

$\mathcal{D}$ is refuted by a CSC iff $\mathcal{D} + [\emptyset, \exists_1\text{-IND}_\eta^{\mathrm{R}-}] \vdash \bot$.

Logical Characterisation

Definition

Γ set of formulas, define

$$\frac{\varphi(0) \quad \varphi(x) \to \varphi(s(x))}{\varphi(\eta)} \ \Gamma\text{-IND}_{\eta}^{\mathrm{R}-}$$

where $\varphi(x) \in \Gamma$.

Definition

T theory, R inference rule, define $[T, R] = T + \{\varphi \mid T \vdash \Gamma, \Gamma/\varphi \in R\}$.

Theorem

$\mathcal{D}$ is refuted by a CSC iff $\mathcal{D} + [\emptyset, \exists_1\text{-IND}_{\eta}^{\mathrm{R}-}] \vdash \bot$.

Proof Sketch.

Induction on clause set ($\forall_1$) in refutation becomes $\exists_1$ induction. □

Outline

Unprovability result

Definition

Define the L_{LA} theory $T = B \cup \{x + y = y + x, x + (y + z) = (x + y) + z\}$.

Definition

Let $k, n, m \in \mathbb{N}$ with $0 < n < m$, define $E_{k,n,m}$ as:

$$n \cdot x + \overline{(m - n)k} = m \cdot x \to x = \overline{k}.$$

For example, $E_{0,1,2}$ is $x + 0 = x + x \to x = 0$.

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$

Corollary

$\mathcal{E}_{k,n,m}(\eta)$ is not refuted by an L_{LA} clause set cycle.

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \rightarrow x = \overline{k}$

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Countermodel $\mathcal{M}$, domain $\{(i,n) \in \mathbb{N} \times \mathbb{Z} \mid i = 0 \text{ implies } n \in \mathbb{N}\}$

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

$$s^{\mathcal{M}}(i,n) = (i, n+1) \qquad p^{\mathcal{M}}((i,n)) = (i, n-1) \text{ if } i > 0$$

$$(i,n) +^{\mathcal{M}} (j,m) = (\max(i,j), n+m)$$

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \not\vdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \not\models E_{k,n,m}$.

We have

$$n \cdot (1, k) +^{\mathcal{M}} \overline{(m-n)k}^{\mathcal{M}} = (1, nk) +^{\mathcal{M}} (0, (m-n)k) = (1, mk) = m \cdot (1, k)$$

but

$$(1, k) \neq (0, k).$$

□

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models T$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\,(L_1 \wedge \cdots \wedge L_n)$

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\,(L_1 \wedge \cdots \wedge L_n)$

Lemma. *If* $\varphi(x)$ *is* $\exists_1$ *then* $\exists N \in \mathbb{N}$, $0, p$*-free components* $\chi_1, \ldots, \chi_l$ *s.t.* $\mathcal{M} \models \varphi(s^N(x)) \leftrightarrow \bigvee_{i=1}^{l} \chi_i(x)$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \not\vdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\,(L_1 \wedge \cdots \wedge L_n)$

Lemma. *If* $\varphi(x)$ *is* $\exists_1$ *then* $\exists N \in \mathbb{N}$, $0, p$*-free components* $\chi_1, \ldots, \chi_l$ *s.t.* $\mathcal{M} \models \varphi(s^N(x)) \leftrightarrow \bigvee_{i=1}^{l} \chi_i(x)$.

Lemma. *If* $0, p$*-free component* $\chi(x)$ *has two solutions in* $\mathbb{N}$ *then* $\exists$ *arith. prog.* $P \subseteq \mathbb{Z}$ *s.t.* $M \models \chi(i, p)$ *for all* $i \geq 1, p \in P$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \not\vdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\, (L_1 \wedge \cdots \wedge L_n)$

Lemma. *If* $\varphi(x)$ *is* $\exists_1$ *then* $\exists N \in \mathbb{N}$, $0, p$*-free components* $\chi_1, \ldots, \chi_l$ *s.t.* $\mathcal{M} \models \varphi(s^N(x)) \leftrightarrow \bigvee_{i=1}^{l} \chi_i(x)$.

Lemma. *If* $0, p$*-free component* $\chi(x)$ *has two solutions in* $\mathbb{N}$ *then* $\exists$ *arith. prog.* $P \subseteq \mathbb{Z}$ *s.t.* $M \models \chi(i, p)$ *for all* $i \geq 1, p \in P$.

Assume $\mathcal{M} \models \varphi(0)$ and $\mathcal{M} \models \varphi(x) \to \varphi(s(x))$. Then $\mathcal{M} \models \varphi((0, n))$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \not\vdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\,(L_1 \wedge \cdots \wedge L_n)$

Lemma. *If* $\varphi(x)$ *is* $\exists_1$ *then* $\exists N \in \mathbb{N}$, $0, p$*-free components* $\chi_1, \ldots, \chi_l$ *s.t.* $\mathcal{M} \models \varphi(s^N(x)) \leftrightarrow \bigvee_{i=1}^{l} \chi_i(x)$.

Lemma. *If* $0, p$*-free component* $\chi(x)$ *has two solutions in* $\mathbb{N}$ *then* $\exists$ *arith. prog.* $P \subseteq \mathbb{Z}$ *s.t.* $M \models \chi(i, p)$ *for all* $i \geq 1, p \in P$.

Assume $\mathcal{M} \models \varphi(0)$ and $\mathcal{M} \models \varphi(x) \to \varphi(s(x))$. Then $\mathcal{M} \models \varphi((0, n))$.
So $\exists l$ s.t. $\chi_l(x)$ has two solutions in $\mathbb{N}$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \to x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\,(L_1 \wedge \cdots \wedge L_n)$

Lemma. *If* $\varphi(x)$ *is* $\exists_1$ *then* $\exists N \in \mathbb{N}$, $0, p$*-free components* $\chi_1, \ldots, \chi_l$ *s.t.* $\mathcal{M} \models \varphi(s^N(x)) \leftrightarrow \bigvee_{i=1}^{l} \chi_i(x)$.

Lemma. *If* $0, p$*-free component* $\chi(x)$ *has two solutions in* $\mathbb{N}$ *then* $\exists$ *arith. prog.* $P \subseteq \mathbb{Z}$ *s.t.* $M \models \chi(i, p)$ *for all* $i \geq 1, p \in P$.

Assume $\mathcal{M} \models \varphi(0)$ and $\mathcal{M} \models \varphi(x) \to \varphi(s(x))$. Then $\mathcal{M} \models \varphi((0, n))$. So $\exists I$ s.t. $\chi_I(x)$ has two solutions in $\mathbb{N}$. So $\mathcal{M} \models \chi_I((i, p))$ for all $i \geq 1, p \in P$.

□

Unprovability result: proof

Theorem (H, Vierling '22)

$T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$, *i.e.*, $n \cdot x + \overline{(m-n)k} = m \cdot x \rightarrow x = \overline{k}$

Proof.

Claim: $\mathcal{M} \models \exists_1\text{-IND}^-$.

Definition. *Component* $\exists \vec{x}\,(L_1 \wedge \cdots \wedge L_n)$

Lemma. *If* $\varphi(x)$ *is* $\exists_1$ *then* $\exists N \in \mathbb{N}$, $0, p$*-free components* $\chi_1, \ldots, \chi_l$ *s.t.* $\mathcal{M} \models \varphi(s^N(x)) \leftrightarrow \bigvee_{i=1}^{l} \chi_i(x)$.

Lemma. *If* $0, p$*-free component* $\chi(x)$ *has two solutions in* $\mathbb{N}$ *then* $\exists$ *arith. prog.* $P \subseteq \mathbb{Z}$ *s.t.* $M \models \chi(i, p)$ *for all* $i \geq 1, p \in P$.

Assume $\mathcal{M} \models \varphi(0)$ and $\mathcal{M} \models \varphi(x) \rightarrow \varphi(s(x))$. Then $\mathcal{M} \models \varphi((0, n))$. So $\exists l$ s.t. $\chi_l(x)$ has two solutions in $\mathbb{N}$. So $\mathcal{M} \models \chi_l((i, p))$ for all $i \geq 1, p \in P$. To prove $\varphi((i, n))$, use sufficiently small (i, p) as basis. □

Outline

Conclusion

Classification of practical methods by means of mathematical logic

Conclusion

Classification of practical methods by means of mathematical logic

- Gauging strength of a method

Conclusion

Classification of practical methods by means of mathematical logic

- Gauging strength of a method
- Independence results for unlimited time and memory

Conclusion

Classification of practical methods by means of mathematical logic

- Gauging strength of a method
- Independence results for unlimited time and memory

Overall:

- A general picture of methods starts to emerge
  (sorted along increasing complexity of induction formulas)

Conclusion

Classification of practical methods by means of mathematical logic

- Gauging strength of a method
- Independence results for unlimited time and memory

Overall:

- A general picture of methods starts to emerge
  (sorted along increasing complexity of induction formulas)
- Techniques for analysing (new) practical methods

Conclusion

Classification of practical methods by means of mathematical logic

- Gauging strength of a method
- Independence results for unlimited time and memory

Overall:

- A general picture of methods starts to emerge
  (sorted along increasing complexity of induction formulas)
- Techniques for analysing (new) practical methods
  - Proof-theoretic (proof translations)

Conclusion

Classification of practical methods by means of mathematical logic

- Gauging strength of a method
- Independence results for unlimited time and memory

Overall:

- A general picture of methods starts to emerge
  (sorted along increasing complexity of induction formulas)
- Techniques for analysing (new) practical methods
  - Proof-theoretic (proof translations)
  - Model-theoretic for independence results (model constructions)

Future Work

- Consolidate results

Future Work

- Consolidate results
- Additional methods:
  term rewriting, Cruanes' calculus, rippling, recursion analysis, ...

Future Work

- Consolidate results
- Additional methods:
  term rewriting, Cruanes' calculus, rippling, recursion analysis, ...
- Relationship to software verification

Future Work

- Consolidate results
- Additional methods:
  term rewriting, Cruanes' calculus, rippling, recursion analysis, ...
- Relationship to software verification
- Theories of inductive data types

Future Work

- Consolidate results
- Additional methods:
  term rewriting, Cruanes' calculus, rippling, recursion analysis, ...
- Relationship to software verification
- Theories of inductive data types
- Deskolemisation: conservativity, complexity, ...

Future Work

- Consolidate results
- Additional methods:
  term rewriting, Cruanes' calculus, rippling, recursion analysis, ...
- Relationship to software verification
- Theories of inductive data types
- Deskolemisation: conservativity, complexity, ...
- Analyticity

Future Work

- Consolidate results
- Additional methods:
  term rewriting, Cruanes' calculus, rippling, recursion analysis, ...
- Relationship to software verification
- Theories of inductive data types
- Deskolemisation: conservativity, complexity, ...
- Analyticity
- Does theoretical understanding help to design better methods?