Arithmetical theories and the automation of induction

Stefan Hetzl   Jannik Vierling

Institute of Discrete Mathematics and Geometry
TU Wien, Austria

42ème Journées sur les Arithmétiques Faibles
Karlovassi, Greece
September 25, 2023

Motivation: two areas

- Mathematical logic: theories of arithmetic
- Computer science: automated inductive theorem proving

Motivation: two areas

- Mathematical logic: theories of arithmetic
- Computer science: automated inductive theorem proving
  - Find proof by induction automatically
    History in computer science dating back to the 1970ies

Motivation: two areas

- Mathematical logic: theories of arithmetic
- Computer science: automated inductive theorem proving
  - Find proof by induction automatically
    History in computer science dating back to the 1970ies
  - Difference to first-order theorem proving (validity):
    No cut-elimination theorem

Motivation: two areas

- Mathematical logic: theories of arithmetic
- Computer science: automated inductive theorem proving
  - Find proof by induction automatically
    History in computer science dating back to the 1970ies
  - Difference to first-order theorem proving (validity):
    No cut-elimination theorem
  - Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...

Motivation: two areas

- Mathematical logic: theories of arithmetic
- Computer science: automated inductive theorem proving
  - Find proof by induction automatically
    History in computer science dating back to the 1970ies
  - Difference to first-order theorem proving (validity):
    No cut-elimination theorem
  - Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
  - Typical problems:
    $\forall x \forall y \; x + y = y + x$
    $\forall l \; \mathrm{len}(\mathrm{rev}(l)) = \mathrm{len}(l)$

Motivation: two areas

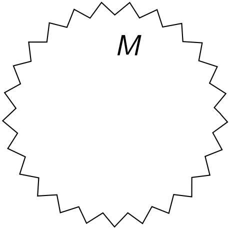
- Mathematical logic: theories of arithmetic
- Computer science: automated inductive theorem proving
  - Find proof by induction automatically
    History in computer science dating back to the 1970ies
  - Difference to first-order theorem proving (validity):
    No cut-elimination theorem
  - Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
  - Typical problems:
    $\forall x \forall y \; x + y = y + x$
    $\forall l \; \mathrm{len}(\mathrm{rev}(l)) = \mathrm{len}(l)$
  - Empirical evaluation of implementations

Motivation: building a connection

- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results

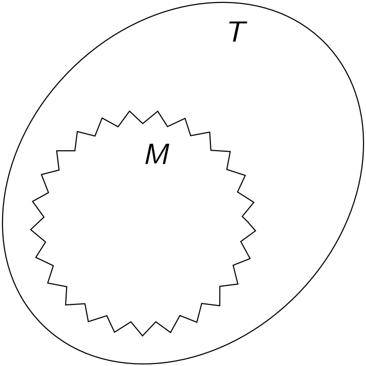
Motivation: building a connection

- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results



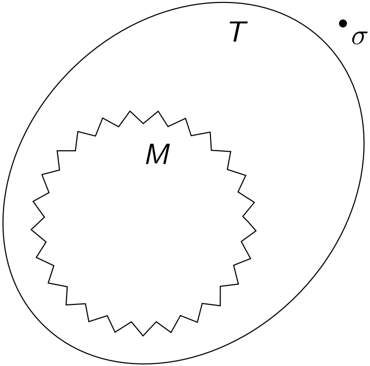
Motivation: building a connection

- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results



Motivation: building a connection

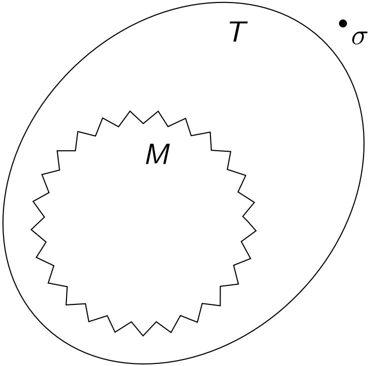
- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results



Motivation: building a connection

- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results

- Straightforward results,
  e.g., $T = \mathrm{PA}$, $\sigma = \mathrm{Con}_{\mathrm{PA}}$

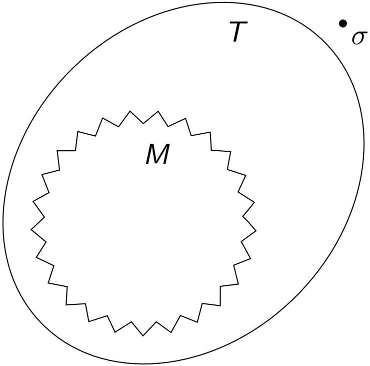


Motivation: building a connection

- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results

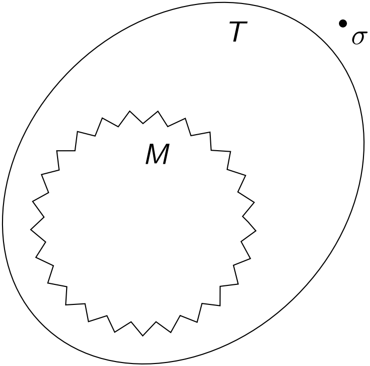
- Straightforward results,
  e.g., $T = \mathrm{PA}$, $\sigma = \mathrm{Con}_{\mathrm{PA}}$

- **Practically meaningful** independence results



Motivation: building a connection

- Use mathematical logic in order to:
  1. classify strength of methods
  2. obtain independence results
- Straightforward results,
  e.g., $T = \mathrm{PA}$, $\sigma = \mathrm{Con}_{\mathrm{PA}}$
- **Practically meaningful** independence results



▶ Weak arithmetical theories

Specifics of this setting

- Often: M and T parametric in language / basic axioms

Specifics of this setting

- Often: M and T parametric in language / basic axioms
- Coding does not play a role

Specifics of this setting

- Often: M and T parametric in language / basic axioms
- Coding does not play a role
- Not only numbers, also: lists, trees, etc.

Specifics of this setting

- Often: M and T parametric in language / basic axioms
- Coding does not play a role
- Not only numbers, also: lists, trees, etc.
- Bounded quantifiers are not distinguished

Specifics of this setting

- Often: M and T parametric in language / basic axioms
- Coding does not play a role
- Not only numbers, also: lists, trees, etc.
- Bounded quantifiers are not distinguished
- Sometimes: idiosyncracies of method M reflected in T

Outline

Induction on Literals

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

Induction on Literals

- **Definition.** A *literal* is an atom or a negated atom.
- **Definition.** The induction axiom $I_x\varphi(x, \overline{z})$ is

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

- **Definition.** For a set of formulas Γ define

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

- In particular: Literal-IND

Induction on Literals

- **Definition.** A *literal* is an atom or a negated atom.
- **Definition.** The induction axiom $I_x\varphi(x,\overline{z})$ is

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

- **Definition.** For a set of formulas Γ define

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

- In particular: Literal-IND
- **Observation.** For natural axioms A in language $L = \{0, s, p, +\}$:

$$A + \text{Literal-IND} \equiv A + \text{Open-IND}.$$

Saturation theorem proving

- Standard technique for automated theorem proving in FOL

Saturation theorem proving

- Standard technique for automated theorem proving in FOL
- **Definition.** *Clause* is a formula $\bigvee_{i=1}^{n} L_i$ where L_i literal.

Saturation theorem proving

- Standard technique for automated theorem proving in FOL
- **Definition.** *Clause* is a formula $\bigvee_{i=1}^{n} L_i$ where L_i literal.
- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules.
- **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.** *Clause* is a formula $\bigvee_{i=1}^{n} L_i$ where L_i literal.
- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules.
- **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 $\overline{L'}$.
- **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 $\mathcal{C}^0 = \mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \ldots \longrightarrow \mathcal{C}^\omega$.

Saturation theorem proving

- Standard technique for automated theorem proving in FOL
- **Definition.** *Clause* is a formula $\bigvee_{i=1}^{n} L_i$ where L_i literal.
- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules.
- **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 $\overline{L'}$.
- **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 $\bot \in \mathcal{C}^\omega$

Adding explicit induction axioms

- Adding induction:

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

Adding explicit induction axioms

- Adding induction:

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

- 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, $L(a)$ variable-free

Adding explicit induction axioms

- Adding induction:

$$\frac{}{\mathsf{CNF}(\mathsf{sk}^{\exists}(I_x\varphi(x)))}$$

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

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

a constant symbol, $L(x)$ literal, $L(a)$ variable-free

- **Example.** $\mathcal{S} + \mathrm{SCIND}$ refutes

$$\{x+0=0, x+s(y)=s(x+y), c+(c+c)\neq(c+c)+c\}$$

Characterisation and independence result

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

Characterisation and independence result

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

- **Definition.**

  Let $T = \{\ 0 \neq s(x), s(x) = s(y) \rightarrow x = y,$
  $E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x)), \neg E(c) \wedge \neg O(c)\}$.

- **Theorem.** T + Literal-IND is consistent.

Characterisation and independence result

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

- **Definition.**

  Let $T = \{\ 0 \neq s(x), s(x) = s(y) \rightarrow x = y,$
  $E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x)), \neg E(c) \wedge \neg O(c)\}.$

- **Theorem.** $T + \text{Literal-IND}$ is consistent.

- **Corollary.** If $\mathcal{S}$ sound saturation system, then $\mathcal{S} + \mathrm{SCIND}$ does not refute T.

Outline

Open induction

- **Theorem.** [Shoenfield '58] Over

$$
\begin{aligned}
s(x) &\neq 0 & x + 0 &= x \\
p(0) &= 0 & x + s(y) &= s(x + y) \\
p(s(x)) &= x
\end{aligned}
$$

  open induction is equivalent to

$$
\begin{aligned}
x + y &= y + x & x &= 0 \vee x = s(p(x)) \\
(x + y) + z &= x + (y + z) & x + y &= x + z \to y = z
\end{aligned}
$$

- Shepherdson '60ies: systematic study

Open induction for lists

- **Definition.** Let $L = \{\mathrm{nil}, \mathrm{cons}, \frown\}$, let $T =$

$$\begin{aligned}
\mathrm{nil} &\neq \mathrm{cons}(x, X) \\
\mathrm{cons}(x, X) = \mathrm{cons}(y, Y) &\to x = y \wedge X = Y \\
\mathrm{nil} \frown Y &= Y \\
\mathrm{cons}(x, X) \frown Y &= \mathrm{cons}(x, X \frown Y)
\end{aligned}$$

Open induction for lists

- **Definition.** Let $L = \{\mathrm{nil}, \mathrm{cons}, \frown\}$, let $T =$

$$\mathrm{nil} \neq \mathrm{cons}(x, X)$$
$$\mathrm{cons}(x, X) = \mathrm{cons}(y, Y) \rightarrow x = y \wedge X = Y$$
$$\mathrm{nil} \frown Y = Y$$
$$\mathrm{cons}(x, X) \frown Y = \mathrm{cons}(x, X \frown Y)$$

- **Theorem.** [H, Vierling]

$$T + \mathit{Open}\text{-}\mathrm{IND} \nvdash Y \frown X = Z \frown X \rightarrow Y = Z.$$

Open Questions

- **Project.** Systematic picture of subsystems of open induction
  - atomic, literal, clause, dual clause, open induction
  - for numbers (in various signatures)
  - for lists, trees, . . .

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 $\mathcal{D}(\eta) \models \mathcal{C}(\eta)$.
- 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 $\mathcal{D}(\eta) \models \mathcal{C}(\eta)$.
- Variants
- **Example.** CSC refutes Even/Odd example

Logical characterisation

- **Definition.** Γ set of formulas, define

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

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

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

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

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 $\mathcal{D} + [\emptyset, \exists_1\text{-IND}_{\eta}^{\mathrm{R}-}] \vdash \bot$.

Independence result

- **Definition.** $L_{\mathrm{LA}} = \{0, s, p, +\}$

  $$\begin{aligned} T = \{ \ & s(0) \neq 0, p(0) = 0, p(s(x)) = x, \\ & x + 0 = x, x + s(y) = s(x + y), \\ & x + y = y + x, x + (y + z) = (x + y) + z \ \} \end{aligned}$$

Independence result

- **Definition.** $L_{\mathrm{LA}} = \{0, s, p, +\}$

  $$\begin{aligned} T = \{ \; & s(0) \neq 0, p(0) = 0, p(s(x)) = x, \\ & x + 0 = x, x + s(y) = s(x + y), \\ & x + y = y + x, x + (y + z) = (x + y) + z \; \} \end{aligned}$$

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

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

Independence result

- **Definition.** $L_{\mathrm{LA}} = \{0, s, p, +\}$

  $$T = \{\ s(0) \neq 0, p(0) = 0, p(s(x)) = x,\\ x + 0 = x, x + s(y) = s(x + y),\\ x + y = y + x, x + (y + z) = (x + y) + z\ \}$$

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

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

- **Example.** $x + 0 = x + x \rightarrow x = 0$ $(E_{0,1,2})$

Independence result

- **Definition.** $L_{\mathrm{LA}} = \{0, s, p, +\}$
  $$T = \{\ s(0) \neq 0, p(0) = 0, p(s(x)) = x, \\ x + 0 = x, x + s(y) = s(x + y), \\ x + y = y + x, x + (y + z) = (x + y) + z\ \}$$
- **Definition.** Let $k, n, m \in \mathbb{N}$ with $0 < n < m$, define $E_{k,n,m}$ as:
  $$n \cdot x + \overline{(m - n)k} = m \cdot x \rightarrow x = \overline{k}$$
- **Example.** $x + 0 = x + x \rightarrow x = 0$ ($E_{0,1,2}$)
- **Theorem.** $T + \exists_1\text{-IND}^- \nvdash E_{k,n,m}$
- **Corollary.** $\mathcal{E}_{k,n,m}(\eta)$ is not refuted by an L_{LA} clause set cycle.

Independence result: proof

Theorem

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

Independence result: proof

Theorem

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

Proof.

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

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

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

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

□

Open Questions

- Iterations of CSCs ... nested applications of $\exists_1\text{-IND}^{\mathsf{R}-}_{\eta}$
- **Theorem.** For every $k \in \mathbb{N}$ there is a clause set $\mathcal{C}_k$ which is refuted by $[\emptyset, \exists_1\text{-IND}^{\mathsf{R}-}]_{k+1}$ but not by $[\emptyset, \exists_1\text{-IND}^{\mathsf{R}-}]_k$
  Proof Sketch. Totality of a suitable function f_k.

Open Questions

- Iterations of CSCs ... nested applications of $\exists_1\text{-IND}_{\eta}^{\mathsf{R}-}$
- **Theorem.** For every $k \in \mathbb{N}$ there is a clause set $\mathcal{C}_k$ which is refuted by $[\emptyset, \exists_1\text{-IND}^{\mathsf{R}-}]_{k+1}$ but not by $[\emptyset, \exists_1\text{-IND}^{\mathsf{R}-}]_k$
  Proof Sketch. Totality of a suitable function f_k.
- **Question.** Is $T + \exists_1\text{-IND}^{-}$ $\forall_2$-conservative over $T + \exists_1\text{-IND}^{\mathsf{R}-}$?

Open Questions

- Iterations of CSCs ... nested applications of $\exists_1$-$\mathrm{IND}^{\mathrm{R}-}_{\eta}$
- **Theorem.** For every $k \in \mathbb{N}$ there is a clause set $\mathcal{C}_k$ which is refuted by $[\emptyset, \exists_1\text{-}\mathrm{IND}^{\mathrm{R}-}]_{k+1}$ but not by $[\emptyset, \exists_1\text{-}\mathrm{IND}^{\mathrm{R}-}]_k$
  Proof Sketch. Totality of a suitable function f_k.
- **Question.** Is $T + \exists_1\text{-}\mathrm{IND}^{-}$ $\forall_2$-conservative over $T + \exists_1\text{-}\mathrm{IND}^{\mathrm{R}-}$?
- **Definition.** Let $T_0 = \{\ s(0) \neq 0, p(0) = 0, p(s(x)) = x,$
  $x + 0 = x, x + s(y) = s(x + y)\}.$
- **Conjecture.** $T_0 + \exists_1\text{-}\mathrm{IND}^{-} \nvdash x + (x + x) = (x + x) + x.$

Open Questions

- Iterations of CSCs ... nested applications of $\exists_1$-$\mathrm{IND}^{\mathrm{R}-}_{\eta}$
- **Theorem.** For every $k \in \mathbb{N}$ there is a clause set $\mathcal{C}_k$ which is refuted by $[\emptyset, \exists_1\text{-}\mathrm{IND}^{\mathrm{R}-}]_{k+1}$ but not by $[\emptyset, \exists_1\text{-}\mathrm{IND}^{\mathrm{R}-}]_k$
  Proof Sketch. Totality of a suitable function f_k.
- **Question.** Is $T + \exists_1\text{-}\mathrm{IND}^{-}$ $\forall_2$-conservative over $T + \exists_1\text{-}\mathrm{IND}^{\mathrm{R}-}$?
- **Definition.** Let $T_0 = \{\ s(0) \neq 0, p(0) = 0, p(s(x)) = x,$
  $x + 0 = x, x + s(y) = s(x + y)\}.$
- **Conjecture.** $T_0 + \exists_1\text{-}\mathrm{IND}^{-} \nvdash x + (x + x) = (x + x) + x.$
  (Note that $T_0 + \text{Literal-IND} \vdash x + (x + x) = (x + x) + x.$)
  (Would yield corollary on CSCs)

Conclusion and future work

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - Independence result for T

Conclusion and future work

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - Independence result for T
- Logical foundations of automated inductive theorem proving
  - Clarify relationship between methods, challenge problems
  - New questions / problems about (weak) arithemtical theories

Conclusion and future work

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - Independence result for T
- Logical foundations of automated inductive theorem proving
  - Clarify relationship between methods, challenge problems
  - New questions / problems about (weak) arithemtical theories

Future Work:

- Analyse further methods: term rewriting, theory exploration, ...

Conclusion and future work

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - Independence result for T

- Logical foundations of automated inductive theorem proving
  - Clarify relationship between methods, challenge problems
  - New questions / problems about (weak) arithemtical theories

Future Work:

- Analyse further methods: term rewriting, theory exploration, ...

- Towards a systematic picture: which method solves which problem?

Conclusion and future work

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - Independence result for T

- Logical foundations of automated inductive theorem proving
  - Clarify relationship between methods, challenge problems
  - New questions / problems about (weak) arithemtical theories

Future Work:

- Analyse further methods: term rewriting, theory exploration, ...
- Towards a systematic picture: which method solves which problem?
- (Weak) theories of inductive datatypes: lists, trees, etc.

Conclusion and future work

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - Independence result for T

- Logical foundations of automated inductive theorem proving
  - Clarify relationship between methods, challenge problems
  - New questions / problems about (weak) arithemtical theories

Future Work:

- Analyse further methods: term rewriting, theory exploration, ...
- Towards a systematic picture: which method solves which problem?
- (Weak) theories of inductive datatypes: lists, trees, etc.
- Analyticity

Thank you!

Stefan Hetzl and Jannik Vierling.
Unprovability results for clause set cycles.
Theoretical Computer Science, 935, 21–46, 2022.

Stefan Hetzl and Jannik Vierling.
Induction and Skolemization in saturation theorem proving.
Annals of Pure and Applied Logic, 174(1):103167, 2023.

Jannik Vierling.
The limits of automated inductive theorem provers.
Ph.D. thesis, TU Wien, Austria.