Gauging the strength of inductive theorem provers

Stefan Hetzl

Institute of Discrete Mathematics and Geometry
TU Wien, Austria

5th Workshop on Automated (Co)inductive Theorem Proving

Nancy, France

July 2, 2024

Automated inductive theorem proving

- History in computer science dating back to the 1970ies

Automated inductive theorem proving

- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...

Automated inductive theorem proving

- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
- No (full) cut-elimination theorem
  - Need to invent new formulas

Automated inductive theorem proving

- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
- No (full) cut-elimination theorem
  - Need to invent new formulas
- (Usually) no completeness theorem
  - Every method proves a different set of theorems.

Automated inductive theorem proving

- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, ...
- No (full) cut-elimination theorem
  - Need to invent new formulas
- (Usually) no completeness theorem
  - Every method proves a different set of theorems.
- Empirical evaluation of implementations

Automated inductive theorem proving

- History in computer science dating back to the 1970ies
- Methods: recursion analysis, term rewriting, rippling, extensions of saturation-based provers, cyclic proofs, theory exploration, . . .
- No (full) cut-elimination theorem
  - Need to invent new formulas
- (Usually) no completeness theorem
  - Every method proves a different set of theorems.
- Empirical evaluation of implementations

- **Given method M, which theorems can M prove?**

What can mathematical logic contribute?

- Rich landscape of inductive theories and knowledge about them
  - various induction schemes
  - various induction rules
  - with/without parameters
  - iterations of rules
  - ...

What can mathematical logic contribute?

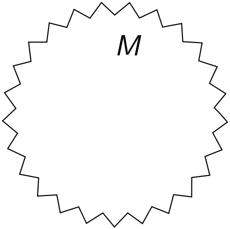
- Rich landscape of inductive theories and knowledge about them
  - various induction schemes
  - various induction rules
  - with/without parameters
  - iterations of rules
  - ...
- **Definition.** A *theory* T is a set of sentences which is deductively closed, i.e., $T \vdash A$ and $A \models B$ implies $T \vdash B$.
- Theory usually specified by set of axioms, e.g., Peano arithmetic (PA) axiomatised by Q and first-order induction scheme

What can mathematical logic contribute?

- Rich landscape of inductive theories and knowledge about them
  - various induction schemes
  - various induction rules
  - with/without parameters
  - iterations of rules
  - ...
- **Definition.** A *theory* T is a set of sentences which is deductively closed, i.e., $T \vdash A$ and $A \models B$ implies $T \vdash B$.
- Theory usually specified by set of axioms, e.g., Peano arithmetic (PA) axiomatised by Q and first-order induction scheme
- $T \vdash \sigma$ iff there is a T proof of σ.
- $T \nvdash \sigma$ iff there is a model $\mathcal{M} \models T$ s.t. $\mathcal{M} \not\models \sigma$.

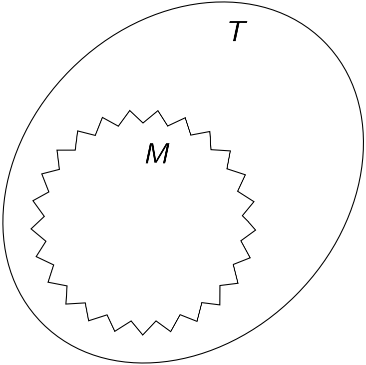
The typical situation

- Given method M for inductive theorem proving . . .



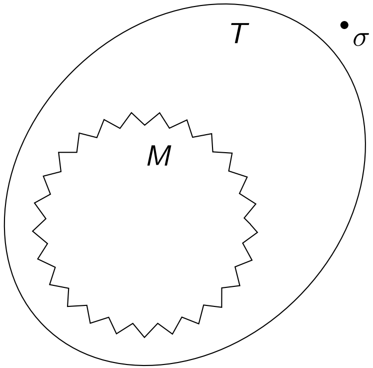
The typical situation

- Given method M for inductive theorem proving ...
- ... find theory T s.t.
  $M \vdash \varphi$ implies $T \vdash \varphi$



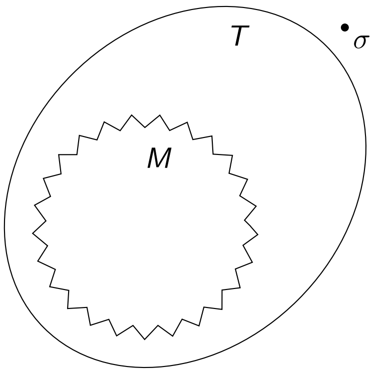
The typical situation

- Given method M for inductive theorem proving ...
- ... find theory T s.t.
  $M \vdash \varphi$ implies $T \vdash \varphi$
- ... and a sentence σ s.t.
  $T \nvdash \sigma$



The typical situation

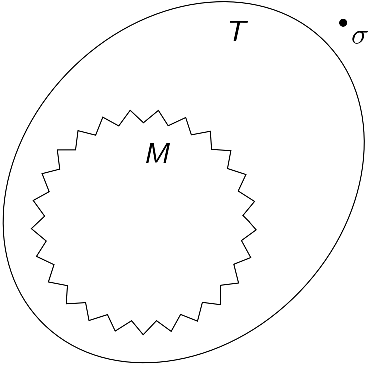
- Given method M for inductive theorem proving ...
- ... find theory T s.t.
  $M \vdash \varphi$ implies $T \vdash \varphi$
- ... and a sentence σ s.t.
  $T \nvdash \sigma$
- Straightforward results,
  e.g., $T = \mathrm{PA}$, $\sigma = \mathrm{Con}_{\mathrm{PA}}$



The typical situation

- Given method M for inductive theorem proving ...
- ... find theory T s.t.
  $M \vdash \varphi$ implies $T \vdash \varphi$
- ... and a sentence σ s.t.
  $T \nvdash \sigma$
- Straightforward results,
  e.g., $T = \mathrm{PA}$, $\sigma = \mathrm{Con}_{\mathrm{PA}}$
- **Practically meaningful** unprovability results

▶ Challenge problems



Outline

Induction in saturation theorem proving

- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules (for adding clauses to the current clause set).

Induction in saturation theorem proving

- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules (for adding clauses to the current clause set).
- **Definition.** Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all all n-ary rules $\rho \in \mathcal{S}$ and all $C_1, \ldots, C_n \in \mathcal{C}$: $\rho(C_1, \ldots, C_n) \subseteq \mathcal{C}$. Given $\mathcal{C}$, compute closure by $\mathcal{C}^0 = \mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \ldots \longrightarrow \mathcal{C}^\omega$.

Induction in saturation theorem proving

- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules (for adding clauses to the current clause set).
- **Definition.** Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all all n-ary rules $\rho \in \mathcal{S}$ and all $C_1, \ldots, C_n \in \mathcal{C}$: $\rho(C_1, \ldots, C_n) \subseteq \mathcal{C}$. 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 $\bot \in \mathcal{C}^\omega$

Induction in saturation theorem proving

- **Definition.** *Saturation system* $\mathcal{S}$ is a set of rules (for adding clauses to the current clause set).
- **Definition.** Clause set $\mathcal{C}$ *closed* under $\mathcal{S}$ if for all all n-ary rules $\rho \in \mathcal{S}$ and all $C_1, \ldots, C_n \in \mathcal{C}$: $\rho(C_1, \ldots, C_n) \subseteq \mathcal{C}$. 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 induction to $\mathcal{S}$:

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

where $\mathrm{sk}^\exists$ is Skolemisation, $I_x\varphi(x)$ is the induction axiom for $\varphi(x)$:

$$\varphi(0) \land \forall x\,(\varphi(x) \to \varphi(s(x))) \to \forall x\,\varphi(x)$$

Single clause induction

- **Example.** 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, $L(a)$ variable-free

Single clause induction

- **Example.** 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

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

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

Single clause induction

- **Example.** 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

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

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

Theorem

$\mathcal{S}$ sound saturation system, $\mathcal{C}$ clause set. If $\mathcal{S} + \mathrm{SCIND}$ refutes $\mathcal{C}$ then the theory $\mathcal{C} + \mathrm{Literal}(L)\text{-}\mathrm{IND}$ is inconsistent.

Proof Sketch.

Essentially by a proof translation (incl. Skolemisation). □

Challenge Problems (1/2)

Challenge Problem (Even/Odd)

$$
\begin{aligned}
\text{Language: } & 0/0, s/1, E/1, O/1 \\
\text{Axioms: } & 0 \neq s(x), s(x) = s(y) \rightarrow x = y, \\
& E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x)) \\
\text{Goal: } & \forall x\,(E(x) \vee O(x))
\end{aligned}
$$

Challenge Problems (1/2)

Challenge Problem (Even/Odd)

$$\begin{aligned} \text{Language: } & 0/0, s/1, E/1, O/1 \\ \text{Axioms: } & 0 \neq s(x), s(x) = s(y) \rightarrow x = y, \\ & E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x)) \\ \text{Goal: } & \forall x\,(E(x) \lor O(x)) \end{aligned}$$

Theorem (Vierling 2024)

Even/Odd is not provable by induction on literals.

Challenge Problems (1/2)

Challenge Problem (Even/Odd)

$$\begin{aligned}
\text{Language: } & 0/0, s/1, E/1, O/1 \\
\text{Axioms: } & 0 \neq s(x), s(x) = s(y) \rightarrow x = y, \\
& E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x)) \\
\text{Goal: } & \forall x\, (E(x) \vee O(x))
\end{aligned}$$

Theorem (Vierling 2024)

Even/Odd is not provable by induction on literals.

Proof Sketch.

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

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

□

Challenge Problems (1/2)

Challenge Problem (Even/Odd)

$$\begin{aligned} \text{Language: } & 0/0, s/1, E/1, O/1 \\ \text{Axioms: } & 0 \neq s(x), s(x) = s(y) \rightarrow x = y, \\ & E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x)) \\ \text{Goal: } & \forall x\,(E(x) \vee O(x)) \end{aligned}$$

Theorem (Vierling 2024)

Even/Odd is not provable by induction on literals.

Corollary

$\mathcal{S}$ sound saturation system. Then $\mathcal{S} + \mathrm{SCIND}$ does not prove Even/Odd.

Challenge Problems (1/2)

Challenge Problem (Even/Odd)

Language: $0/0, s/1, E/1, O/1$

Axioms: $0 \neq s(x), s(x) = s(y) \rightarrow x = y,$
$E(0), E(x) \rightarrow O(s(x)), O(x) \rightarrow E(s(x))$

Goal: $\forall x\,(E(x) \vee O(x))$

Theorem (Vierling 2024)

Even/Odd is not provable by induction on literals.

Corollary

$\mathcal{S}$ sound saturation system. Then $\mathcal{S} + \mathrm{SCIND}$ does not prove Even/Odd.

Remark

Vampire with multi-clause induction proves Even/Odd.

Challenge Problems (2/2)

Challenge Problem (C_2)

Language: $0/0, s/1, p/1, +/2$

Axioms: $s(x) \neq 0, p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y)$

Goal: $\forall x \, \forall y \, (x + x = y + y \to x = y)$

Challenge Problems (2/2)

Challenge Problem (C_2)

Language: $0/0, s/1, p/1, +/2$

Axioms: $s(x) \neq 0, p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y)$

Goal: $\forall x \, \forall y \, (x + x = y + y \to x = y)$

Challenge Problem ($D_{2,1}$)

Language and axioms as in C_2

Goal: $\forall x \, \forall y \, s(x + x) \neq y + y$

Challenge Problems (2/2)

Challenge Problem (C_2)

Language: $0/0, s/1, p/1, +/2$

Axioms: $s(x) \neq 0, p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y)$

Goal: $\forall x \, \forall y \, (x + x = y + y \rightarrow x = y)$

Challenge Problem ($D_{2,1}$)

Language and axioms as in C_2

Goal: $\forall x \, \forall y \, s(x + x) \neq y + y$

Theorem (Shoenfield 1958)

Open induction does not prove C_2 nor $D_{2,1}$.

Challenge Problems (2/2)

Challenge Problem (C_2)

Language: $0/0, s/1, p/1, +/2$

Axioms: $s(x) \neq 0, p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y)$

Goal: $\forall x \forall y \, (x + x = y + y \to x = y)$

Challenge Problem ($D_{2,1}$)

Language and axioms as in C_2

Goal: $\forall x \forall y \, s(x + x) \neq y + y$

Theorem (Shoenfield 1958)

Open induction does not prove C_2 nor $D_{2,1}$.

Corollary

$\mathcal{S}$ sound saturation system. Then $\mathcal{S} + \mathrm{SCIND}$ does not prove C_2 nor $D_{2,1}$

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 proves Even/Odd.

Logical Characterisation

- Induction with and without parameters

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

Logical Characterisation

- Induction with and without parameters

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

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

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

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

Logical Characterisation

- Induction with and without parameters

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

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

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

  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

- Induction with and without parameters

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

- **Definition.** 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 (H, Vierling 2022)

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

Challenge problem

Challenge Problem ($E_{0,1,2}$)

Language: $0/0, s/1, p/1, +/2$

Axioms: $0 \neq s(x), p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y),$
$x + y = y + x, x + (y + z) = (x + y) + z$

Goal: $\forall x\,(x + 0 = x + x \rightarrow x = 0)$

Challenge problem

Challenge Problem ($E_{0,1,2}$)

Language: $0/0, s/1, p/1, +/2$

Axioms: $0 \neq s(x), p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y),$
$x + y = y + x, x + (y + z) = (x + y) + z$

Goal: $\forall x\, (x + 0 = x + x \rightarrow x = 0)$

Theorem (H, Vierling 2022)

$\exists_1$-IND$^-$ *does not prove* $E_{0,1,2}$.

Challenge problem

Challenge Problem ($E_{0,1,2}$)

Language: $0/0, s/1, p/1, +/2$

Axioms: $0 \neq s(x), p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y),$
$x + y = y + x, x + (y + z) = (x + y) + z$

Goal: $\forall x\,(x + 0 = x + x \rightarrow x = 0)$

Theorem (H, Vierling 2022)

$\exists_1$-IND$^-$ *does not prove* $E_{0,1,2}$.

Proof Sketch.

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

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

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

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

Challenge problem

Challenge Problem ($E_{0,1,2}$)

Language: $0/0, s/1, p/1, +/2$

Axioms: $0 \neq s(x), p(0) = 0, p(s(x)) = x, x + 0 = x, x + s(y) = s(x + y),$
$x + y = y + x, x + (y + z) = (x + y) + z$

Goal: $\forall x \, (x + 0 = x + x \rightarrow x = 0)$

Theorem (H, Vierling 2022)

$\exists_1$-IND^- *does not prove* $E_{0,1,2}$.

Corollary

CSC does not prove $E_{0,1,2}$.

Outline

Cancellation

- The natural numbers satisfy:
  - Left cancellation C_{l}: $x + y = x + z \rightarrow y = z$
  - Right cancellation C_{r}: $y + x = z + x \rightarrow y = z$
  - **Observation.** Open induction proves C_{l} and C_{r}.

Cancellation

- The natural numbers satisfy:
  - Left cancellation C_{l}: $x + y = x + z \rightarrow y = z$
  - Right cancellation C_{r}: $y + x = z + x \rightarrow y = z$
  - **Observation.** Open induction proves C_{l} and C_{r}.

- Sequences with nil, cons and concatenation $\frown$ satsify:
  - left cancellation: $X \frown Y = X \frown Z \rightarrow Y = Z$
  - right cancellation: $Y \frown X = Z \frown X \rightarrow Y = Z$

Cancellation

- The natural numbers satisfy:
  - Left cancellation C_{l}: $x + y = x + z \to y = z$
  - Right cancellation C_{r}: $y + x = z + x \to y = z$
  - **Observation.** Open induction proves C_{l} and C_{r}.

- Sequences with nil, cons and concatenation $\frown$ satsify:
  - left cancellation: $X \frown Y = X \frown Z \to Y = Z$
  - right cancellation: $Y \frown X = Z \frown X \to Y = Z$

- Provable by list induction?

$$\forall \bar{z} \left(\varphi(\mathrm{nil}, \bar{z}) \land \forall x \, \forall Y \left(\varphi(Y, \bar{z}) \to \varphi(\mathrm{cons}(x, Y), \bar{z})\right) \to \forall X \, \varphi(X, \bar{z})\right)$$

Cancellation

- The natural numbers satisfy:
  - Left cancellation C_{l}: $x + y = x + z \to y = z$
  - Right cancellation C_{r}: $y + x = z + x \to y = z$
  - **Observation.** Open induction proves C_{l} and C_{r}.
- Sequences with nil, cons and concatenation $\frown$ satsify:
  - left cancellation: $X \frown Y = X \frown Z \to Y = Z$
  - right cancellation: $Y \frown X = Z \frown X \to Y = Z$
- Provable by list induction?

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

- **Observation.** Open (list) induction proves left cancellation for lists.

Challenge Problem (1/2)

Challenge Problem (Right List Cancellation)

Language: nil : list, cons : $\iota \times$ list $\rightarrow$ list, $\frown$: list $\times$ list $\rightarrow$ list

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

$\mathrm{nil} \frown Y = Y, \mathrm{cons}(x, X) \frown Y = \mathrm{cons}(x, X \frown Y)$

Goal: $\forall X \, \forall Y \, \forall Z \, (Y \frown X = Z \frown X \rightarrow Y = Z)$

Challenge Problem (1/2)

Challenge Problem (Right List Cancellation)

Language: $\mathrm{nil} : \mathrm{list}, \mathrm{cons} : \iota \times \mathrm{list} \to \mathrm{list}, \frown : \mathrm{list} \times \mathrm{list} \to \mathrm{list}$

Axioms: $\mathrm{nil} \neq \mathrm{cons}(x, X), \mathrm{cons}(x, X) = \mathrm{cons}(y, Y) \to x = y \land X = Y,$

$\mathrm{nil} \frown Y = Y, \mathrm{cons}(x, X) \frown Y = \mathrm{cons}(x, X \frown Y)$

Goal: $\forall X \, \forall Y \, \forall Z \, (Y \frown X = Z \frown X \to Y = Z)$

Theorem (H, Vierling 2024)

Open induction does not prove Right List Cancellation.

Challenge Problem (1/2)

Challenge Problem (Right List Cancellation)

$$\begin{aligned}
\text{Language: } & \mathrm{nil} : \mathrm{list}, \mathrm{cons} : \iota \times \mathrm{list} \to \mathrm{list}, \frown : \mathrm{list} \times \mathrm{list} \to \mathrm{list} \\
\text{Axioms: } & \mathrm{nil} \neq \mathrm{cons}(x, X), \mathrm{cons}(x, X) = \mathrm{cons}(y, Y) \to x = y \land X = Y, \\
& \mathrm{nil} \frown Y = Y, \mathrm{cons}(x, X) \frown Y = \mathrm{cons}(x, X \frown Y) \\
\text{Goal: } & \forall X \, \forall Y \, \forall Z \, (Y \frown X = Z \frown X \to Y = Z)
\end{aligned}$$

Theorem (H, Vierling 2024)

Open induction does not prove Right List Cancellation.

Proof Sketch.

Idea: $(a) \frown (a, a, a, \ldots) = (a, a, a, \ldots) = \mathrm{nil} \frown (a, a, a, \ldots)$ but $(a) \neq \mathit{nil}$

$\square$

Challenge Problem (1/2)

Challenge Problem (Right List Cancellation)

Language: $\text{nil} : \text{list}, \text{cons} : \iota \times \text{list} \to \text{list}, \frown : \text{list} \times \text{list} \to \text{list}$

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

$\text{nil} \frown Y = Y, \text{cons}(x, X) \frown Y = \text{cons}(x, X \frown Y)$

Goal: $\forall X \, \forall Y \, \forall Z \, (Y \frown X = Z \frown X \to Y = Z)$

Theorem (H, Vierling 2024)

Open induction does not prove Right List Cancellation.

Proof Sketch.

Idea: $(a) \frown (a, a, a, \ldots) = (a, a, a, \ldots) = \text{nil} \frown (a, a, a, \ldots)$ but $(a) \neq nil$

$\mathcal{M}$: (a certain set of) sequences of length $< \omega^3$ $\square$

Challenge Problem (1/2)

Challenge Problem (Right List Cancellation)

Language: $\mathrm{nil} : \mathrm{list}, \mathrm{cons} : \iota \times \mathrm{list} \to \mathrm{list}, \frown : \mathrm{list} \times \mathrm{list} \to \mathrm{list}$

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

Goal: $\forall X \, \forall Y \, \forall Z \, (Y \frown X = Z \frown X \to Y = Z)$

Theorem (H, Vierling 2024)

Open induction does not prove Right List Cancellation.

Remark

$\forall_1$*-induction proves Right List Cancellation.*

Challenge Problem (2/2)

Challenge Problem (Right List Decomposition)

Language: $\mathrm{nil} : \mathrm{list}, \mathrm{cons} : \iota \times \mathrm{list} \to \mathrm{list}, \frown : \mathrm{list} \times \mathrm{list} \to \mathrm{list}$

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

Goal: $\forall X \, (X = \mathrm{nil} \lor \exists Y \, \exists z \, X = Y \frown \mathrm{cons}(z, \mathrm{nil}))$

Challenge Problem (2/2)

Challenge Problem (Right List Decomposition)

Language: nil : list, cons : $\iota \times$ list $\to$ list, $\frown$: list $\times$ list $\to$ list

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

Goal: $\forall X\, (X = \mathrm{nil} \vee \exists Y\, \exists z\, X = Y \frown \mathrm{cons}(z, \mathrm{nil}))$

Theorem (H, Vierling 2024)

Open induction does not prove Right List Decomposition.

Challenge Problem (2/2)

Challenge Problem (Right List Decomposition)

Language: $\mathrm{nil} : \mathrm{list}, \mathrm{cons} : \iota \times \mathrm{list} \to \mathrm{list}, \frown : \mathrm{list} \times \mathrm{list} \to \mathrm{list}$

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

Goal: $\forall X\, (X = \mathrm{nil} \vee \exists Y\, \exists z\, X = Y \frown \mathrm{cons}(z, \mathrm{nil}))$

Theorem (H, Vierling 2024)

Open induction does not prove Right List Decomposition.

Observation

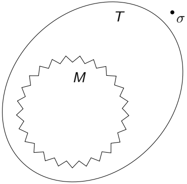
Open induction proves **Left** *List Decomposition.*

$$\forall X\, (X = \mathrm{nil} \vee \exists Y\, \exists z\, X = \mathrm{cons}(z, \mathrm{nil}) \frown Y)$$

Outline

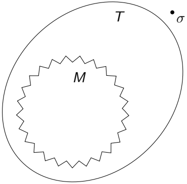
Conclusion

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - (Practically relevant) statement σ unprovable in T
- T overapproximates M



Conclusion

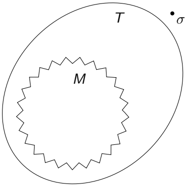
- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - (Practically relevant) statement σ unprovable in T
- T overapproximates M



- Logical foundations of automated inductive theorem proving
- Challenge problems

Conclusion

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - (Practically relevant) statement σ unprovable in T
- T overapproximates M

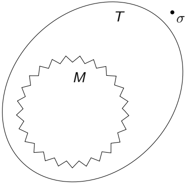


- Logical foundations of automated inductive theorem proving
- Challenge problems

- Designers of automated inductive theorem provers:
  Where is my prover in this landscape of theories?

Conclusion

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - (Practically relevant) statement σ unprovable in T
- T overapproximates M



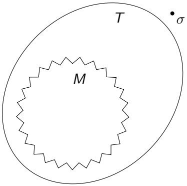
- Logical foundations of automated inductive theorem proving
- Challenge problems

- Designers of automated inductive theorem provers:
  Where is my prover in this landscape of theories?

Future Work:

Conclusion

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - (Practically relevant) statement σ unprovable in T
- T overapproximates M



- Logical foundations of automated inductive theorem proving
- Challenge problems

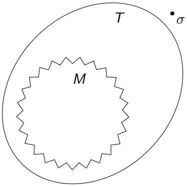
- Designers of automated inductive theorem provers:
  Where is my prover in this landscape of theories?

Future Work:

- More on other inductive datatypes: lists, trees, etc.

Conclusion

- Strategy for analysing method M:
  - Find upper bound T for strength of M
  - (Practically relevant) statement σ unprovable in T
- T overapproximates M



- Logical foundations of automated inductive theorem proving
- Challenge problems

- Designers of automated inductive theorem provers:
  Where is my prover in this landscape of theories?

Future Work:

- More on other 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.

- Stefan Hetzl and Jannik Vierling.
  Quantifier-free induction for lists.
  Archive for Mathematical Logic, published online, 2024.

- Jannik Vierling.
  The limits of automated inductive theorem provers.
  PhD thesis, TU Wien, Austria, 2024.