Universal truth of operator statements via ideal membership

Clemens Hofstadler, Clemens G. Raab, Georg Regensburger
Institute for Mathematics, University of Kassel

Seminar Algebra and Discrete Mathematics
Linz, Austria, December 1, 2022

UNIKASSEL VERSITÄT

FWF Der Wissenschaftsfonds.

Introduction

Goal Proving statements about matrices/linear operators automatically and efficiently!

Introduction

Goal Proving statements about matrices/linear operators **automatically** and **efficiently**!

$$\begin{array}{c}\text{statement about}\\ \text{operators}\end{array}\Bigg\downarrow\begin{array}{c}\text{statement}\\ \text{is true}\end{array}$$

Introduction

Goal Proving statements about matrices/linear operators automatically and efficiently!

$$\begin{tikzcd}
\text{statement about operators} \arrow[r, rightsquigarrow] \arrow[d] & \text{first-order formula} \\
\text{statement is true} \arrow[r, rightsquigarrow] & \text{semantic consequence}
\end{tikzcd}$$

Introduction

Goal Proving statements about matrices/linear operators automatically and efficiently!

$$\begin{tikzcd}
\text{statement about operators} \arrow[r, rightsquigarrow] \arrow[d] & \text{first-order formula} \arrow[d, "\text{formal computation}"'] \\
\text{statement is true} \arrow[r, rightsquigarrow] & \text{semantic consequence}
\end{tikzcd}$$

Introduction

Goal Proving statements about matrices/linear operators
automatically and efficiently!

$$
\begin{tikzcd}
\text{statement about operators} \arrow[r, squiggly] \arrow[d] & \text{first-order formula} \arrow[r, leftrightarrow] \arrow[d, "\text{formal computation}"'] & \text{noncommutative ideal memberships} \\
\text{statement is true} \arrow[r, squiggly] & \text{semantic consequence} \arrow[r, leftrightarrow] & \text{memberships hold}
\end{tikzcd}
$$

Introduction

Goal Proving statements about matrices/linear operators **automatically** and **efficiently**!

$$
\begin{tikzcd}
\text{statement about operators} \arrow[r, squiggly] \arrow[d] & \text{first-order formula} \arrow[r, leftrightarrow] \arrow[d, "\text{formal computation}"'] & \text{noncommutative ideal memberships} \arrow[d, "\text{efficient computation}"'] \\
\text{statement is true} \arrow[r, squiggly] & \text{semantic consequence} \arrow[r, leftrightarrow] & \text{memberships hold}
\end{tikzcd}
$$

Modelling operator statements
via
many-sorted first-order logic

Many-sorted first-order logic

- First-order logic + sorts
- Same expressiveness as unsorted FO logic
- Computational advantages (sorts reduce # of expressions)
- We use sorts to model domains and codomains

Linear operator statements

“Statements”

Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \to P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)

Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \rightarrow P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \to P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

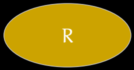
"Linear operators"

Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \rightarrow P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

"Linear operators"

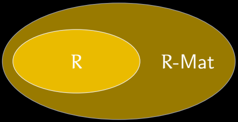


Linear operator statements

“Statements”

- $\bigwedge_i S_i = T_i \to P = Q$ (Raab, Regensburger, Hossein Poor, ’19/’21)
- First-order logic with equality as only predicate (today)

“Linear operators”

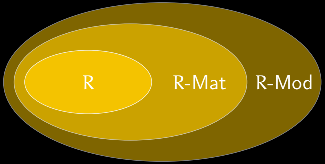


Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \rightarrow P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

"Linear operators"

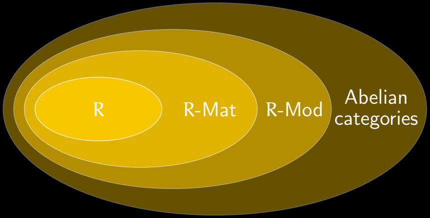


Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \to P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

"Linear operators"

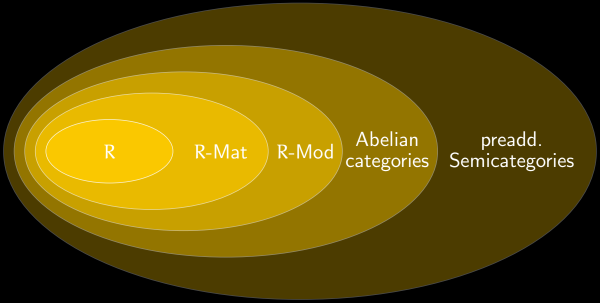


Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \to P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

"Linear operators"

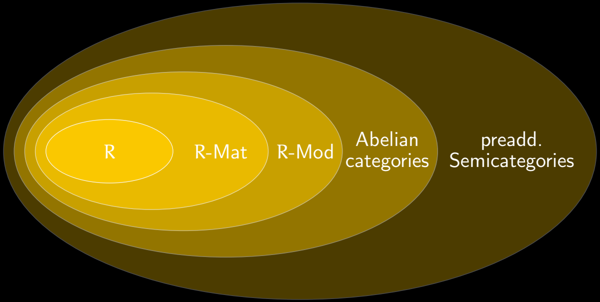


Linear operator statements

"Statements"

- $\bigwedge_i S_i = T_i \to P = Q$ (Raab, Regensburger, Hossein Poor, '19/'21)
- First-order logic with equality as only predicate (today)

"Linear operators"



Semicategory = Category − identity morphisms

Signature

$$\mathbf{Var} = \{x_1, x_2, \dots\} \dots \text{variables}$$

A signature is a tuple $\Sigma = (O, C, F, \sigma)$ consisting of

- $O \dots$ object symbols $\qquad (u, v) \in O \times O \dots$ sort
- $C \dots$ constant symbols with $0_{u,v} \in C$ for $u, v \in O$;
- $F \dots$ function symbols with $-, +, \cdot \in F$;
- $\sigma \dots$ sort function assigning all symbols their sort

Syntax

Terms of sort (u, v)

- variables $x \in \mathbf{Var}$ with $\sigma(x) = (u, v)$
- constants $c \in C$ with $\sigma(c) = (u, v)$
- $f(t_1, \ldots t_n)$ with $\sigma(t_i) = (u_i, v_i)$ and

$$\sigma(f) = (u_1, v_1) \times \cdots \times (u_n, v_n) \to (u, v)$$

Syntax

Terms of sort (u, v)

- variables $x \in \mathbf{Var}$ with $\sigma(x) = (u, v)$
- constants $c \in C$ with $\sigma(c) = (u, v)$
- $f(t_1, \ldots t_n)$ with $\sigma(t_i) = (u_i, v_i)$ and
$$\sigma(f) = (u_1, v_1) \times \cdots \times (u_n, v_n) \to (u, v)$$

Formulas

- $s \approx t$ with terms s, t where $\sigma(s) = \sigma(t)$
- $\neg\, \varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$
- $\exists\, x : \varphi$, $\forall\, x : \varphi$

Syntax

Fix $\quad \sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$$x + y$$

Syntax

Fix $\quad \sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$$\cancel{x + y}$$

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$

Syntax

Fix $\quad \sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

Syntax

Fix $\quad \sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

$x \approx y$

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

$\cancel{x \approx y}$

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

$\cancel{x \approx y}$

$\forall x : x \approx 0 \cdot (c + c)$

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

$\cancel{x \approx y}$

$\forall x : x \approx 0 \cdot (c + c)$ arithmetic sentence

Syntax

Fix $\sigma(x) = (u, v), \qquad \sigma(y) = (u, w), \qquad \sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

$\cancel{x \approx y}$

$\forall x : x \approx 0 \cdot (c + c)$ arithmetic sentence

$$(c \not\approx 0 \wedge c + (-c) \approx 0) \to -c \not\approx 0$$

Syntax

Fix $\sigma(x) = (u, v)$, $\sigma(y) = (u, w)$, $\sigma(c) = (u, w)$

$\cancel{x + y}$

$(c + y) \cdot 0_{w,u}$ term of sort (w, w)

$0_{w,v} \cdot (c + c)$ ground term of sort (u, v)

$\cancel{x \approx y}$

$\forall x : x \approx 0 \cdot (c + c)$ arithmetic sentence

$(c \not\approx 0 \wedge c + (-c) \approx 0) \rightarrow -c \not\approx 0$

arithmetic ground sentence

Axioms of semicategories

$$
\begin{aligned}
\mathcal{A}_{u,u',v,v'} = \{ & \\
& \forall x^{(v',v)}, y^{(u',v')}, z^{(u,u')} && : \quad x \cdot (y \cdot z) \approx (x \cdot y) \cdot z, \\
& \forall x^{(u,v)}, y^{(u,v)}, z^{(u,v)} && : \quad x + (y + z) \approx (x + y) + z, \\
& \forall x^{(u,v)} && : \quad x + 0_{u,v} \approx x, \\
& \forall x^{(u,v)} && : \quad x + (-x) \approx 0_{u,v}, \\
& \forall x^{(u,v)}, y^{(u,v)} && : \quad x + y \approx y + x, \\
& \forall x^{(v',v)}, y^{(u,v')}, z^{(u,v')} && : \quad x \cdot (y + z) \approx x \cdot y + x \cdot z, \\
& \forall x^{(v',v)}, y^{(v',v)}, z^{(u,v')} && : \quad (x + y) \cdot z \approx x \cdot z + y \cdot z \\
\} &
\end{aligned}
$$

Axioms $\mathcal{A}$ of semicategories are then

$$\mathcal{A} = \bigcup_{u,u',v,v' \in O} \mathcal{A}_{u,u',v,v'}$$

Semantics

Formulas are true or false w.r.t. interpretation $\mathcal{I}$

Interpretations are as in classical FO logic, except that they respect the sorts.

Semantics

Formulas are true or false w.r.t. interpretation $\mathcal{I}$

Interpretations are as in classical FO logic, except that they respect the sorts.

For example:

$$\mathcal{I}(s \approx t) = \top \quad \text{iff} \quad \mathcal{I}(s) = \mathcal{I}(t) \text{ as elements in the domain}$$

Semantics

Formulas are true or false w.r.t. interpretation $\mathcal{I}$

Interpretations are as in classical FO logic, except that they respect the sorts.

For example:

$$\mathcal{I}(s \approx t) = \top \quad \text{iff} \quad \mathcal{I}(s) = \mathcal{I}(t) \text{ as elements in the domain}$$

$$\varphi \text{ valid} \quad \text{iff} \quad \mathcal{I}(\varphi) = \top \text{ for all } \mathcal{I}$$

$$\Psi \models \varphi \text{ (sem. consequence)} \quad \text{iff} \quad \mathcal{I}(\Psi) = \top \Rightarrow \mathcal{I}(\varphi) = \top \text{ for all } \mathcal{I}$$

Semantics

Formulas are true or false w.r.t. interpretation $\mathcal{I}$

Interpretations are as in classical FO logic, except that they respect the sorts.

For example:

$$\mathcal{I}(s \approx t) = \top \quad \text{iff} \quad \mathcal{I}(s) = \mathcal{I}(t) \text{ as elements in the domain}$$

$$\varphi \text{ valid} \quad \text{iff} \quad \mathcal{I}(\varphi) = \top \text{ for all } \mathcal{I}$$

$$\Psi \models \varphi \text{ (sem. consequence)} \quad \text{iff} \quad \mathcal{I}(\Psi) = \top \Rightarrow \mathcal{I}(\varphi) = \top \text{ for all } \mathcal{I}$$

Universal truth of operator statement $\equiv$ $\mathcal{A} \models \varphi$

Formal computation

Goal Prove semantic consequence via syntactic operations

Classically using some deductive system (e.g., Sequent calculus, Resolution, etc.)

Formal computation

Goal Prove semantic consequence via syntactic operations

Classically using some deductive system (e.g., Sequent calculus, Resolution, etc.)

$\Psi \vdash \varphi$ (syn. consequence) iff φ can be derived from Ψ syntactically

Formal computation

Goal Prove semantic consequence via syntactic operations

Classically using some deductive system (e.g., Sequent calculus, Resolution, etc.)

$\Psi \vdash \varphi$ (syn. consequence) iff φ can be derived from Ψ syntactically

Theorem $\Psi \vDash \varphi$ iff $\Psi \vdash \varphi$

Formal computation

Goal Prove semantic consequence via syntactic operations

Classically using some deductive system (e.g., Sequent calculus, Resolution, etc.)

$\Psi \vdash \varphi$ (syn. consequence) iff φ can be derived from Ψ syntactically

Correctness

Theorem $\Psi \vDash \varphi$ iff $\Psi \vdash \varphi$

Formal computation

Goal Prove semantic consequence via syntactic operations

Classically using some deductive system (e.g., Sequent calculus, Resolution, etc.)

$\Psi \vdash \varphi$ (syn. consequence) iff φ can be derived from Ψ syntactically

Theorem

$$\begin{tikzcd} \Psi \vDash \varphi \arrow[rr, bend right, "\text{Completeness}"'] & \text{iff} & \Psi \vdash \varphi \arrow[ll, bend right, "\text{Correctness}"'] \end{tikzcd}$$

Formal computation

Goal Prove semantic consequence via syntactic operations

Classically using some deductive system (e.g., Sequent calculus, Resolution, etc.)

$\Psi \vdash \varphi$ (syn. consequence) iff φ can be derived from Ψ syntactically

Theorem

$$\begin{tikzcd} \Psi \vDash \varphi \arrow[r, phantom, "\text{iff}"] \arrow[r, bend right, "\text{Completeness}"'] & \Psi \vdash \varphi \arrow[l, bend right, "\text{Correctness}"'] \end{tikzcd}$$

Part II Derive analogous statement for $\Psi = \mathcal{A}$ with polynomial rhs

Expressing universal truth
by
polynomial ideal memberships

Interlude: Noncommutative polynomials

Noncommutative polynomials $=$ elements in free algebra $\mathbb{Z}\langle X\rangle$

$$= \sum_{i=1}^{d} c_i \cdot x_{i,1} \ldots x_{i,k_i}$$

Interlude: Noncommutative polynomials

Noncommutative polynomials $=$ elements in free algebra $\mathbb{Z}\langle X\rangle$

$$= \sum_{i=1}^{d} c_i \cdot x_{i,1} \ldots x_{i,k_i}$$

For $F \subseteq \mathbb{Z}\langle X\rangle$,

$$(F) = \left\{ \sum a_i f_i b_i \mid a_i, b_i \in \mathbb{Z}\langle X\rangle,\ f_i \in F \right\}$$

Interlude: Noncommutative polynomials

Noncommutative polynomials $=$ elements in free algebra $\mathbb{Z}\langle X\rangle$

$$= \sum_{i=1}^{d} c_i \cdot x_{i,1} \dots x_{i,k_i}$$

For $F \subseteq \mathbb{Z}\langle X\rangle$,

$$(F) = \left\{ \sum a_i f_i b_i \mid a_i, b_i \in \mathbb{Z}\langle X\rangle,\ f_i \in F \right\}$$

Ideal membership problem $p \overset{?}{\in} (F)$ is semi-decidable

(e.g., using Gröbner bases)

From formulas to polynomials

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$ (Idealisation)

From formulas to polynomials

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$ (Idealisation)

Translating arithmetic ground terms is easy

$$a \cdot (a + 0) + (-b) + c \cdot d \quad \rightsquigarrow \quad aa - b + cd \in \mathbb{Z}\langle a, b, c, d \rangle$$

From formulas to polynomials

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$ (Idealisation)

Translating arithmetic ground terms is easy

$$a \cdot (a + 0) + (-b) + c \cdot d \quad \rightsquigarrow \quad aa - b + cd \in \mathbb{Z}\langle a, b, c, d\rangle$$

...but what about

$$\exists x : a + b \approx x \qquad \rightsquigarrow \quad ?$$

From formulas to polynomials

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$ (Idealisation)

Translating arithmetic ground terms is easy

$$a \cdot (a + 0) + (-b) + c \cdot d \quad \rightsquigarrow \quad aa - b + cd \in \mathbb{Z}\langle a, b, c, d\rangle$$

...but what about

$$\exists x : a + b \approx x \quad \rightsquigarrow \quad ?$$

$$f(a, b) \approx f(b, a) \quad \rightsquigarrow \quad ?$$

From formulas to polynomials

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$ (Idealisation)

Translating arithmetic ground terms is easy

$$a \cdot (a + 0) + (-b) + c \cdot d \quad \rightsquigarrow \quad aa - b + cd \in \mathbb{Z}\langle a, b, c, d\rangle$$

...but what about

$$\exists x : a + b \approx x \quad \rightsquigarrow \quad ? \qquad \leftarrow \text{Herbrand's theorem}$$

$$f(a, b) \approx f(b, a) \quad \rightsquigarrow \quad ? \qquad \leftarrow \text{Ackermann's reduction}$$

Herbrand's theorem

Goal Reduce validity of formula to validity of ground sentence
$\rightsquigarrow$ eliminate quantifiers

Herbrand's theorem

Goal Reduce validity of formula to validity of ground sentence
$\rightsquigarrow$ eliminate quantifiers

Herbrand normal form : $\exists x_1 \ldots x_n : \varphi^*$ with φ^* quantifier-free

Herbrand's theorem

Goal Reduce validity of formula to validity of ground sentence
$\rightsquigarrow$ eliminate quantifiers

Herbrand normal form : $\exists x_1 \ldots x_n : \varphi^*$ with φ^* quantifier-free

Herbrand expansion : $H(\varphi) = \{$ all ground instances of φ^* $\}$

Herbrand's theorem

Goal Reduce validity of formula to validity of ground sentence
$\rightsquigarrow$ eliminate quantifiers

Herbrand normal form : $\exists x_1 \dots x_n : \varphi^*$ with φ^* quantifier-free

Herbrand expansion : $H(\varphi) = \{$ all ground instances of $\varphi^* \}$

Herbrand's theorem Let φ be in Herbrand normal form. Then φ is valid if and only if there exist $\varphi_1, \dots, \varphi_k \in H(\varphi)$ such that $\varphi_1 \vee \dots \vee \varphi_k$ is valid.

Herbrand's theorem

Goal Reduce validity of formula to validity of ground sentence

$\rightsquigarrow$ eliminate quantifiers

Herbrand normal form : $\exists x_1 \ldots x_n : \varphi^*$ with φ^* quantifier-free

Herbrand expansion : $H(\varphi) = \{$ all ground instances of $\varphi^* \}$

Herbrand's theorem Let φ be in Herbrand normal form. Then φ is valid if and only if there exist $\varphi_1, \ldots, \varphi_k \in H(\varphi)$ such that $\varphi_1 \vee \cdots \vee \varphi_k$ is valid.

Corollary $\mathcal{A} \vDash \varphi$ iff $\mathcal{A} \vDash \varphi_1 \vee \cdots \vee \varphi_k$.

Herbrandisation

Goal Transform formula φ into Herbrand normal form

Herbrandisation

Goal Transform formula φ into Herbrand normal form

1. Bind any free variables by universal quantifier
2. Move all quantifiers to the front
3. Apply the following rules exhaustively

$$\begin{aligned} \forall y : \psi \quad &\rightsquigarrow \quad \psi[y \mapsto c] \\ \exists x_1, \ldots, x_n \forall y : \psi \quad &\rightsquigarrow \quad \exists x_1, \ldots, x_n : \psi[y \mapsto f(x_1, \ldots, x_n)] \end{aligned}$$

where c, f are new with the correct sort

Herbrandisation

Goal Transform formula φ into Herbrand normal form

1. Bind any free variables by universal quantifier
2. Move all quantifiers to the front
3. Apply the following rules exhaustively

$$\begin{aligned} \forall y : \psi \quad &\rightsquigarrow \quad \psi[y \mapsto c] \\ \exists x_1, \dots, x_n \forall y : \psi \quad &\rightsquigarrow \quad \exists x_1, \dots, x_n : \psi[y \mapsto f(x_1, \dots, x_n)] \end{aligned}$$

where c, f are new with the correct sort

Proposition $\mathcal{A} \vDash \varphi \quad \text{iff} \quad \mathcal{A} \vDash \varphi^H$

Ackermann's reduction

Goal Remove function symbol f from quantifier-free formula φ

Ackermann's reduction

Goal Remove function symbol f from quantifier-free formula φ

1. Flatten nested applications of f
2. Replace every instance $f(t_1, \dots, t_n)$ by new constant $c_{f,t_1,\dots,t_n}$. Denote new formula by φ^{flat}.
3. For all $c_{f,s_1,\dots,s_n}$ and $c_{f,t_1,\dots,t_n}$ form functional consistency constraint

$$s_1 \approx t_1 \wedge \dots \wedge s_n \approx t_n \;\rightarrow\; c_{f,s_1,\dots,s_n} \approx c_{f,t_1,\dots,t_n}$$

4. $\varphi^{\text{FC}} \leftarrow$ Conjunction of all functional consistency constraints
5. $\varphi^{\text{Ack}} \leftarrow \left(\varphi^{\text{FC}} \rightarrow \varphi^{\text{flat}}\right)$

Ackermann's reduction

Goal Remove function symbol f from quantifier-free formula φ

1. Flatten nested applications of f
2. Replace every instance $f(t_1, \ldots, t_n)$ by new constant $c_{f,t_1,\ldots,t_n}$. Denote new formula by φ^{flat}.
3. For all $c_{f,s_1,\ldots,s_n}$ and $c_{f,t_1,\ldots,t_n}$ form functional consistency constraint

$$s_1 \approx t_1 \wedge \cdots \wedge s_n \approx t_n \;\rightarrow\; c_{f,s_1,\ldots,s_n} \approx c_{f,t_1,\ldots,t_n}$$

4. $\varphi^{\mathsf{FC}} \leftarrow$ Conjunction of all functional consistency constraints
5. $\varphi^{\mathsf{Ack}} \leftarrow \left(\varphi^{\mathsf{FC}} \rightarrow \varphi^{\mathsf{flat}}\right)$

Theorem φ valid iff φ^{Ack} valid

Ackermann's reduction

Goal Remove function symbol f from quantifier-free formula φ

1. Flatten nested applications of f
2. Replace every instance $f(t_1, \ldots, t_n)$ by new constant $c_{f,t_1,\ldots,t_n}$. Denote new formula by φ^{flat}.
3. For all $c_{f,s_1,\ldots,s_n}$ and $c_{f,t_1,\ldots,t_n}$ form functional consistency constraint

$$s_1 \approx t_1 \wedge \cdots \wedge s_n \approx t_n \ \rightarrow \ c_{f,s_1,\ldots,s_n} \approx c_{f,t_1,\ldots,t_n}$$

4. $\varphi^{\mathsf{FC}} \leftarrow$ Conjunction of all functional consistency constraints
5. $\varphi^{\mathsf{Ack}} \leftarrow \left(\varphi^{\mathsf{FC}} \rightarrow \varphi^{\mathsf{flat}}\right)$

Theorem φ valid iff φ^{Ack} valid

Corollary If non-arithmetic function symbol is removed, then

$$\mathcal{A} \vDash \varphi \qquad \text{iff} \qquad \mathcal{A} \vDash \varphi^{\mathsf{Ack}}$$

Idealisation

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$

Idealisation

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$

Suffices to discuss arithmetic ground sentences

$$\text{arb. formula} \xrightarrow{\text{Herbrand}} \text{ground sent.} \xrightarrow{\text{Ackermann}} \text{arith. ground sent.}$$

Idealisation

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $\mathrm{I}(\varphi)$

Suffices to discuss arithmetic ground sentences

$$\text{arb. formula} \xrightarrow{\text{Herbrand}} \text{ground sent.} \xrightarrow{\text{Ackermann}} \text{arith. ground sent.}$$

Every φ is logically equivalent to a formula of the form

$$\mathrm{CNF}(\varphi) = \bigwedge_i \underbrace{\left(\bigvee_j s_{i,j} \not\approx t_{i,j} \vee \bigvee_k p_{i,k} \approx q_{i,k} \right)}_{\text{clause}}$$

Idealisation

Goal Translate $\mathcal{A} \vDash \varphi$ into polynomial predicate $I(\varphi)$

Suffices to discuss arithmetic ground sentences

$$\text{arb. formula} \xrightarrow{\text{Herbrand}} \text{ground sent.} \xrightarrow{\text{Ackermann}} \text{arith. ground sent.}$$

Every φ is logically equivalent to a formula of the form

$$\mathrm{CNF}(\varphi) = \bigwedge_i \underbrace{\left(\bigvee_j s_{i,j} \not\approx t_{i,j} \vee \bigvee_k p_{i,k} \approx q_{i,k} \right)}_{\text{clause}}$$

Idealisation

clause:

$$I(C_i) \quad :\equiv \quad p_{i,k} - q_{i,k} \in (s_{i,1} - t_{i,1}, \ldots, s_{i,n} - t_{i,n}) \text{ for some } k$$

Idealisation

Goal Translate $\mathcal{A} \models \varphi$ into polynomial predicate $I(\varphi)$

Suffices to discuss arithmetic ground sentences

$$\text{arb. formula} \xrightarrow{\text{Herbrand}} \text{ground sent.} \xrightarrow{\text{Ackermann}} \text{arith. ground sent.}$$

Every φ is logically equivalent to a formula of the form

$$\mathrm{CNF}(\varphi) = \bigwedge_i \underbrace{\left(\bigvee_j s_{i,j} \not\approx t_{i,j} \vee \bigvee_k p_{i,k} \approx q_{i,k} \right)}_{\text{clause}}$$

Idealisation

clause:

$$I(C_i) \quad :\equiv \quad p_{i,k} - q_{i,k} \in (s_{i,1} - t_{i,1}, \ldots, s_{i,n} - t_{i,n}) \text{ for some } k$$

arith. ground sentence: $I(\varphi) \quad :\equiv \quad \bigwedge_{\substack{C \text{ clause} \\ \text{of } \mathrm{CNF}(\varphi)}} I(C)$

Main result

Theorem Let φ be an arithmetic ground sentence. Then

$$\mathcal{A} \vDash \varphi \qquad \text{iff} \qquad \mathrm{I}(\varphi) = \top.$$

Main result

Theorem Let φ be an arithmetic ground sentence. Then

$$\mathcal{A} \models \varphi \qquad \text{iff} \qquad \mathrm{I}(\varphi) = \top.$$

Proof:

"$\Leftarrow$": Reduce to (Raab, Regensburger, Hossein Poor, '19/'21)

Main result

Theorem Let φ be an arithmetic ground sentence. Then

$$\mathcal{A} \vDash \varphi \qquad \text{iff} \qquad \mathrm{I}(\varphi) = \top.$$

Proof:

"$\Leftarrow$": Reduce to (Raab, Regensburger, Hossein Poor, '19/'21)

"$\Rightarrow$": Use $\mathcal{A} \vDash \varphi$ iff $A \vdash \varphi$ and show that sequent rules respect idealisation.

Main result

Theorem Let φ be an arithmetic ground sentence. Then

$$\mathcal{A} \models \varphi \qquad \text{iff} \qquad \mathrm{I}(\varphi) = \top.$$

Proof:

"$\Leftarrow$": Reduce to (Raab, Regensburger, Hossein Poor, '19/'21)

"$\Rightarrow$": Use $\mathcal{A} \models \varphi$ iff $\mathrm{A} \vdash \varphi$ and show that sequent rules respect idealisation.

Advantages

- Axioms $\mathcal{A}$ are treated implicitly.
- Proof is independent of sorts, and thus, holds in all settings.
- Exploit efficient polynomial routines.

Semi-decision procedure

Input: signature Σ, formula φ

Output: $\top$ if and only if $\mathcal{A} \vDash \varphi$; otherwise infinite loop

1. $\varphi^{H} \leftarrow$ Herband normal form of φ
2. $\varphi_1, \varphi_2, \cdots \leftarrow$ an enumeration of $H(\varphi^{H})$
3. $n \leftarrow 1$
4. $\psi_n \leftarrow \bigvee_{i=1}^{n} \varphi_i$
5. $\psi_n^{\mathsf{Ack}} \leftarrow$ remove all non-arithmetic function symbols from ψ_n using Ackermann's reduction.
6. If $I(\psi_n^{\mathsf{Ack}}) = \top$, return $\top$. Otherwise, increase n by 1 and go to step 4.

Semi-decision procedure

Input: signature Σ, formula φ

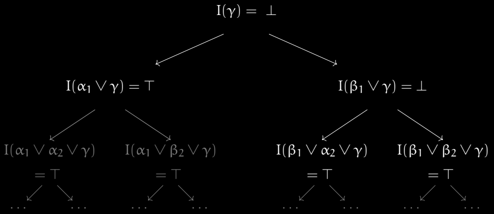
Output: $\top$ if and only if $\mathcal{A} \vDash \varphi$; otherwise infinite loop

1. $\varphi^H \leftarrow$ Herband normal form of φ
2. $\varphi_1, \varphi_2, \cdots \leftarrow$ an enumeration of $H(\varphi^H)$
3. $n \leftarrow 1$
4. $\psi_n \leftarrow \bigvee_{i=1}^{n} \varphi_i$
5. $\psi_n^{\mathsf{Ack}} \leftarrow$ remove all non-arithmetic function symbols from ψ_n using Ackermann's reduction.
6. For $k \leftarrow 1, \ldots, n$

   If $I(\psi_k^{\mathsf{Ack}}) = \top$ can be verified with n operations, return $\top$.
7. Increase n by 1 and go to step 4.

Computational aspects

- Efficiency of the procedure $\approx$ good enumeration of $\mathrm{H}(\varphi^{\mathrm{H}})$
  - Expert knowledge
  - "Polynomial unification"
- Avoid CNF blowup by incremental computation



- Treat simple subformulas separately

Applications

Used to automatically (im)prove statements in the field of

- generalized inverses
  - Moore-Penrose inverses (K. Bernauer)

Applications

(Extract from Handbook of Linear Algebra)

5.7 Pseudo-Inverse

Definitions:

A **Moore–Penrose pseudo-inverse** of a matrix $A \in \mathbb{C}^{m\times n}$ is a matrix $A^\dagger \in \mathbb{C}^{n\times m}$ that satisfies the following four **Penrose** conditions:

$$AA^\dagger A = A; \quad A^\dagger AA^\dagger = A^\dagger; \quad (AA^\dagger)^* = AA^\dagger; \quad (A^\dagger A)^* = A^\dagger A.$$

Facts:

All the following facts except those with a specific reference can be found in [Gra83, pp. 105–141] or [RM71, pp. 44–67].

1. Every $A \in \mathbb{C}^{m\times n}$ has a unique pseudo-inverse $A^\dagger$.
2. If $A \in \mathbb{R}^{m\times n}$, then $A^\dagger$ is real.
3. If $A \in \mathbb{C}^{m\times n}$ of rank r has a full rank decomposition $A = BC$, where $B \in \mathbb{C}^{m\times r}$ and $C \in \mathbb{C}^{r\times n}$, then $A^\dagger$ can be evaluated using $A^\dagger = C^*(B^*AC^*)^{-1}B^*$.
4. [LH95, p. 38] If $A \in \mathbb{C}^{m\times n}$ of rank $r \le \min\{m,n\}$ has an SVD $A = U\Sigma V^*$, then its pseudo-inverse is $A^\dagger = V\Sigma^\dagger U^*$, where

$$\Sigma^\dagger = \operatorname{diag}(1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0) \in \mathbb{R}^{n\times m}.$$

---

10. $(A^\dagger)^* = (A^*)^\dagger$; $(A^\dagger)^\dagger = A$.
11. If A is a nonsingular square matrix, then $A^\dagger = A^{-1}$.
12. If U has orthonormal columns or orthonormal rows, then $U^\dagger = U^*$.
13. If $A = A^*$ and $A = A^2$, then $A^\dagger = A$.
14. $A^\dagger = A^*$ if and only if A^*A is idempotent.
15. If A is normal and k is a positive integer, then $AA^\dagger = A^\dagger A$ and $(A^k)^\dagger = (A^\dagger)^k$.
16. If $U \in \mathbb{C}^{m\times n}$ is of rank n and satisfies $U^\dagger = U^*$, then U has orthonormal columns.
17. If $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices, then $(UAV)^\dagger = V^*A^\dagger U^*$.
18. $A^\dagger = (A^*A)^\dagger A^* = A^*(AA^*)^\dagger$. In particular,

Applications

Used to automatically (im)prove statements in the field of

- Generalized inverses
  - Moore-Penrose inverses (K. Bernauer)

Applications

Used to automatically (im)prove statements in the field of

- Generalized inverses
  - Moore-Penrose inverses (K. Bernauer)
  - Reverse order law (with D. Cvetković-Ilić and J. Milošević)

Applications

Used to automatically (im)prove statements in the field of

- Generalized inverses
  - Moore-Penrose inverses (K. Bernauer)
  - Reverse order law (with D. Cvetković-Ilić and J. Milošević)

- Solvability of systems of equations

Applications

(Milošević, '20)

Theorem 2.2 *Let a_i, b_i, c_i be elements of a ring $\mathcal{R}$ with a unit such that a_i, b_i are regular and $a_i a_i^- c_i b_i^- b_i = c_i$ for $i = \overline{1,3}$. Additionally, let $s = a_2 l_{a_1}, j = a_3 l_{a_1}, m = j l_s, t = r_{b_1} b_2, k = r_{b_1} b_3, n = r_t k, p = a_3 l_{a_2}, q = r_{b_2} b_3$ and $s, j, m, t, k, n, r_m p, q l_n, r_{r_m p} r_m j, q l_k l_n \in \mathcal{R}^-$. The following are equivalent:*

(*i*) *The system of equations* (11) *is consistent.*

(*ii*) *The conditions*

$$
\begin{aligned}
&r_s(c_2 - a_2 a_1^- c_1 b_1^- b_2) l_t = 0\\
&r_{r_m p} r_m j (r_{r_m p} r_m j)^- r_{r_m p} r_m e l_n (q l_n)^- q l_n = r_{r_m p} r_m e l_n\\
&r_m j j^- e l_k l_n (q l_k l_n)^- q l_k l_n = r_m e l_k l_n
\end{aligned}
$$

are satisfied, where $e = c_3 - js^- c_2 t^- k - a_3 a_2^- r_s c_2 t^- k - js^- c_2 l_t b_2^- b_3 - (a_3 - js^- a_2) a_1^- c_1 b_1^- (b_3 - b_2 t^- k)$.

(*iii*) *The conditions*

$$
\begin{aligned}
&r_s(c_2 - a_2 a_1^- c_1 b_1^- b_2) l_t = 0\\
&r_j(c_3 - a_3 a_1^- c_1 b_1^- b_3) l_k = 0\\
&r_m(c_3 - js^- c_2 l_t b_2^- b_3 - (a_3 - js^- a_2) a_1^- c_1 b_1^- b_3) l_k l_n l_{q l_k l_n} = 0\\
&r_{r_{r_m p} r_m j} r_{r_m p} r_m (c_3 - a_3 a_2^- r_s c_2 t^- k - a_3 a_1^- c_1 b_1^- (b_3 - b_2 t^- k)) l_n = 0\\
&r_{r_m p} r_m (c_3 - js^- c_2 t^- k - a_3 a_2^- r_s c_2 t^- k - js^- c_2 l_t b_2^- b_3\\
&-(a_3 - js^- a_2) a_1^- c_1 b_1^- (b_3 - b_2 t^- k)) l_n l_{q l_n} = 0.
\end{aligned}
$$

are satisfied.

In that case the general solution of (11) *is given by* (34), *where*

$$
\begin{aligned}
z_1 = {} & c_1 f + g^- r_s c_2 t^- t + a_1 l_{a_2} (r_m p)^- r_m [e - j(r_{r_m p} r_m j)^- r_{r_m p} r_m e \\
& - j l_{r_{r_m p} r_m j} s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) r_{q l_k l_n} q - (1 - j(r_{r_m p} r_m j)^- r_{r_m p} r_m) e l_k l_n (q l_k l_n)^- q] l_n k^- t \\
& + a_1 a_1^- l_g u_1 t t^- - a_1 l_{a_2} (r_m p)^- r_m p (1 - l_{a_1} s^- a_2) a_1^- u_1 t^- k l_n k^- t, \\
z_2 = {} & (c_1 b_1^- (b_3 - b_2 t^- k) + g^- r_s c_2 t^- k) l_n + a_1 a_3^- c_3 n^- n + a_1 (1 - a_3^- r_m r_j a_3) a_1^- u_2 n^- n \\
& a_1 l_{a_2} (r_m p)^- r_m [e - j(r_{r_m p} r_m j)^- r_{r_m p} r_m e - j l_{r_{r_m p} r_m j} s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) r_{q l_k l_n} q \\
& - (1 - j(r_{r_m p} r_m j)^- r_{r_m p} r_m) e l_k l_n (q l_k l_n)^- q] k^- k l_n + a_1 a_1^- l_g u_1 t^- k l_n \\
& - a_1 l_{a_2} (r_m p)^- r_m p (1 - l_{a_1} s^- a_2) a_1^- u_1 t^- k l_n, \\
z_3 = {} & g c_1 + s s^- c_2 l_t f^- + s (r_{r_m p} r_m j)^- r_{r_m p} r_m e l_n (q l_n)^- r_{b_2} b_1 \\
& + s (l_{r_{r_m p} r_m j} j^- + (1 - j^- r_m j)(r_{r_m p} r_m j)^- r_{r_m p}) r_m e l_k l_n (q l_k l_n)^- r_{b_2} b_1 \\
& + s s^- u_3 r_f b_1^- b_1 - s j^- r_m j l_{r_{r_m p} r_m j} s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) q l_k l_n (q l_k l_n)^- r_{b_2} b_1 \\
& - s (r_{r_m p} r_m j)^- r_{r_m p} r_m j s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) q l_n (q l_n)^- r_{b_2} b_1, \\
z_4 = {} & g c_1 b_1^- (b_3 - b_2 t^- k) l_n + + s s^- c_2 l_t b_2^- b_3 l_n + (g g^- r_s c_2 + s s^- c_2) t^- k l_n + r_s a_2 a_3^- c_3 n^- n \\
& + r_s a_2 (1 - a_3^- r_m r_j a_3) a_1^- u_2 n^- n + s j^- r_m j [s^- a_2 a_3^- c_3 + (s^- a_2 (1 - a_3^- r_m r_j a_3) - j^- a_3) a_1^- u_2] n^- n \\
& + s (1 - j^- r_m j) s^- u_4 n^- n + s (r_{r_m p} r_m j)^- r_{r_m p} r_m e l_n (q l_n)^- r_{b_2} b_1 b_1^- (b_3 - b_2 t^- k) l_n \\
& + s (l_{r_{r_m p} r_m j} j^- + (1 - j^- r_m j)(r_{r_m p} r_m j)^- r_{r_m p}) r_m e l_k l_n (q l_k l_n)^- r_{b_2} b_1 b_1^- (b_3 - b_2 t^- k) l_n \\
& + s s^- u_3 r_f b_1^- (b_3 - b_2 t^- k) l_n \\
& - s j^- r_m j l_{r_{r_m p} r_m j} s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) q l_k l_n (q l_k l_n)^- r_{b_2} b_1 b_1^- (b_3 - b_2 t^- k) l_n \\
& - s (r_{r_m p} r_m j)^- r_{r_m p} r_m j s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) q l_n (q l_n)^- r_{b_2} b_1 b_1^- (b_3 - b_2 t^- k) l_n, \\
z_5 = {} & r_m ((a_3 - j s^- a_2) a_1^- c_1 + j s^- c_2 l_t f^-) + m m^- c_3 b_3^- b_1 + m m^- u_5 b_1^- (1 - b_3 l_k l_n b_3^-) b_1 \\
& + r_m j (r_{r_m p} r_m j)^- r_{r_m p} r_m e l_n (q l_n)^- r_{b_2} b_1 + r_m j l_{r_{r_m p} r_m j} j^- r_m e l_k l_n (q l_k l_n)^- r_{b_2} b_1 + r_m j s^- u_3 r_f b_1^- b_1 \\
& - r_m j l_{r_{r_m p} r_m j} s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) q l_k l_n (q l_k l_n)^- r_{b_2} b_1 \\
& - r_m j (r_{r_m p} r_m j)^- r_{r_m p} r_m j s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) q l_n (q l_n)^- r_{b_2} b_1, \\
z_6 = {} & r_m (a_3 - j s^- a_2) a_1^- c_1 f + r_m a_3 a_2^- r_s c_2 t^- t + r_m j s^- (c_2 l_t f^- f + c_2 t^- t) + m m^- c_3 b_3^- b_2 l_t \\
& + m m^- u_5 b_1^- (1 - b_3 l_k l_n b_3^-) b_2 l_t + m m^- [c_3 b_3^- b_2 t^- + u_5 b_1^- ((1 - b_3 l_k l_n b_3^-) b_2 t^- - b_3 k^-)] k l_n k^- t \\
& + m m^- u_6 t^- (1 - k l_n k^-) t + r_m (a_3 - j s^- a_2) a_1^- l_g u_1 t^- t \\
& + r_m (a_3 - j s^- a_2) a_1^- a_1 l_{a_2} (r_m p)^- r_m [e - j l_{r_{r_m p} r_m j} s^- u_3 b_1^- (1 - b_2 t^- r_{b_1}) r_{q l_k l_n} q \\
& - j (r_{r_m p} r_m j)^- r_{r_m p} r_m e - (1 - j (r_{r_m p} r_m j)^- r_{r_m p} r_m) e l_k l_n (q l_k l_n)^- q] l_n k^- t \\
& - r_m (a_3 - j s^- a_2) a_1^- a_1 l_{a_2} (r_m p)^- r_m p (1 - l_{a_1} s^- a_2) a_1^- u_1 t^- k l_n k^- t,
\end{aligned}
$$

Applications

Used to automatically (im)prove statements in the field of

- Generalized inverses
  - Moore-Penrose inverses (K. Bernauer)
  - Reverse order law (with D. Cvetković-Ilić and J. Milošević)
- Solvability of systems of equations
- Homological algebra

Applications

Used to automatically (im)prove statements in the field of

- Generalized inverses
  - Moore-Penrose inverses (K. Bernauer)
  - Reverse order law (with D. Cvetković-Ilić and J. Milošević)
- Solvability of systems of equations
- Homological algebra
  - Diagram lemmas

Applications

Used to automatically (im)prove statements in the field of

- Generalized inverses
  - Moore-Penrose inverses (K. Bernauer)
  - Reverse order law (with D. Cvetković-Ilić and J. Milošević)
- Solvability of systems of equations
- Homological algebra
  - Diagram lemmas

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\alpha"] & B \arrow[r, "g"] \arrow[d, "\beta"] & C \arrow[r, "h"] \arrow[d, "\gamma"] & D \arrow[r, "j"] \arrow[d, "\delta"] & E \arrow[d, "\varepsilon"] \\ A' \arrow[r, "f'"'] & B' \arrow[r, "g'"'] & C' \arrow[r, "h'"'] & D' \arrow[r, "j'"'] & E' \end{tikzcd}$$

Summary & Outlook

Summary

- Model operator statements via many-sorted logic
- Translate validity of operator statement into finitely many polynomial ideal memberships
- Tools: Herbrand's theorem + Ackermann's reduction

Outlook

- Producing proofs
- (Better) heuristics for finding good instantiations
- More advanced computational techniques (DPLL-style, techniques from SMT)
- Further applications