Undecidability of Dyadic First-Order Logic in Coq

Johannes Hostert[1], Andrej Dudenhefner[1,2], Dominik Kirst[1]

[1]Saarland University

[2]TU Dortmund

ITP 2022, Haifa, August 9th

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

- $+$ is function symbol (of arity 2)
- $=$ is relation symbol (of arity 2)

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

- $+$ is function symbol (of arity 2)
- $=$ is relation symbol (of arity 2)
- **Signature**: collection of non-logical symbols (and their arity)
- Follow existing mechanization in FOL library [Kirst et al., 2022]
  - See Dominik's talk on Friday, 9am @ Coq Workshop

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

- $+$ is function symbol (of arity 2)
- $=$ is relation symbol (of arity 2)
- **Signature**: collection of non-logical symbols (and their arity)
- Follow existing mechanization in FOL library [Kirst et al., 2022]
  - See Dominik's talk on Friday, 9am @ Coq Workshop

Well-known decision problem: The **Entscheidungsproblem**

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

- $+$ is function symbol (of arity 2)
- $=$ is relation symbol (of arity 2)
- **Signature**: collection of non-logical symbols (and their arity)
- Follow existing mechanization in FOL library [Kirst et al., 2022]
  - See Dominik's talk on Friday, 9am @ Coq Workshop

Well-known decision problem: The **Entscheidungsproblem**

- VAL: Is a formula **valid** in all models?
- PRV: Is a formula **provable**?
- SAT: Is a formula **satisfied** in some model?

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

- $+$ is function symbol (of arity 2)
- $=$ is relation symbol (of arity 2)
- **Signature**: collection of non-logical symbols (and their arity)
- Follow existing mechanization in FOL library [Kirst et al., 2022]
    - See Dominik's talk on Friday, 9am @ Coq Workshop

Well-known decision problem: The **Entscheidungsproblem**

- VAL: Is a formula **valid** in all models?
- PRV: Is a formula **provable**?
- SAT: Is a formula **satisfied** in some model?

Classically: $\overline{\mathsf{SAT}}(\neg\varphi) \Leftrightarrow \mathsf{VAL}\,\varphi \Leftrightarrow \mathsf{PRV}_c\,\varphi$

First-Order Logic

Quantifying over individuals: $\forall ab.\, a + b = b + a$

- $+$ is function symbol (of arity 2)
- $=$ is relation symbol (of arity 2)
- **Signature**: collection of non-logical symbols (and their arity)
- Follow existing mechanization in FOL library [Kirst et al., 2022]
  - See Dominik's talk on Friday, 9am @ Coq Workshop

Well-known decision problem: The **Entscheidungsproblem**

- VAL: Is a formula **valid** in all models?
- PRV: Is a formula **provable**?
- SAT: Is a formula **satisfied** in some model?

Classically: $\overline{\mathsf{SAT}}(\neg\varphi) \Leftrightarrow \mathsf{VAL}\,\varphi \Leftrightarrow \mathsf{PRV}_c\,\varphi$
All undecidable in general case [Church, 1936, Turing, 1936]

Mechanized undecidability

- Classical Undecidability
  - Choose model of computation (Turing Machines, λ calculus)
  - Show a problem undecidable (Halting Problem, PCP, ...)

Mechanized undecidability

- Classical Undecidability
  - Choose model of computation (Turing Machines, λ calculus)
  - Show a problem undecidable (Halting Problem, PCP, ...)
  - Build reductions to show more problems undecidable

Mechanized undecidability

- Classical Undecidability
  - Choose model of computation (Turing Machines, λ calculus)
  - Show a problem undecidable (Halting Problem, PCP, ...)
  - Build reductions to show more problems undecidable
  - Use Church-Turing-Thesis to argue computability intuitively

Mechanized undecidability

- Classical Undecidability
  - Choose model of computation (Turing Machines, λ calculus)
  - Show a problem undecidable (Halting Problem, PCP, ...)
  - Build reductions to show more problems undecidable
  - Use Church-Turing-Thesis to argue computability intuitively
- Synthetic Undecidability [Forster, Kirst, and Smolka, 2019, Forster, 2021]
  - In Coq: All definable functions are computable
  - Mechanize many-one reductions without explicit model of computation

Mechanized undecidability

- Classical Undecidability
  - Choose model of computation (Turing Machines, λ calculus)
  - Show a problem undecidable (Halting Problem, PCP, ...)
  - Build reductions to show more problems undecidable
  - Use Church-Turing-Thesis to argue computability intuitively

- Synthetic Undecidability [Forster, Kirst, and Smolka, 2019, Forster, 2021]
  - In Coq: All definable functions are computable
  - Mechanize many-one reductions without explicit model of computation
  - Start reductions at well-known undecidable problem

Mechanized undecidability

- Classical Undecidability
  - Choose model of computation (Turing Machines, λ calculus)
  - Show a problem undecidable (Halting Problem, PCP, ...)
  - Build reductions to show more problems undecidable
  - Use Church-Turing-Thesis to argue computability intuitively
- Synthetic Undecidability [Forster, Kirst, and Smolka, 2019, Forster, 2021]
  - In Coq: All definable functions are computable
  - Mechanize many-one reductions without explicit model of computation
  - Start reductions at well-known undecidable problem
  - Heavily inspired by Synthetic Computability [Richman, 1983, Bauer, 2006]
  - Axiom-free, intuitionistic approach

Variants of the Entscheidungsproblem

We consider VAL, SAT and PRV, subject to restrictions:

Variants of the Entscheidungsproblem

We consider VAL, SAT and PRV, subject to restrictions:

- **Monadic** signature: only unary symbols
- **Dyadic** signature: exactly one binary predicate

Variants of the Entscheidungsproblem

We consider VAL, SAT and PRV, subject to restrictions:

- **Monadic** signature: only unary symbols
- **Dyadic** signature: exactly one binary predicate
- Restricted to **finite** models

Variants of the Entscheidungsproblem

We consider VAL, SAT and PRV, subject to restrictions:

- **Monadic** signature: only unary symbols
- **Dyadic** signature: exactly one binary predicate
- Restricted to **finite** models
- Syntax restricted to $(\forall, \rightarrow, \bot)$-fragment (**small** fragment)
- Syntax restricted to $(\forall, \rightarrow)$-fragment (**without negation**)
- Not considered: small quantifier prefixes (i.e. $\forall\exists^*\forall$)

Classifying FOL Problems

Undecidable | Decidable

Classifying FOL Problems

Undecidable | Decidable

Entscheidungsproblem
[Turing, 1936, Church, 1936]

Classifying FOL Problems

Undecidable

Decidable

PRV, VAL, SAT
[Turing, 1936, Church, 1936]

Classifying FOL Problems

Undecidable

PRV, VAL, SAT
[Turing, 1936, Church, 1936]

Dyadic variant
[Kalmár, 1939]

Decidable

Classifying FOL Problems

Undecidable	Decidable
PRV, VAL, SAT [Turing, 1936, Church, 1936]	
Dyadic variant [Kalmár, 1939]	**Monadic variant** [Löwenheim, 1915]

Classifying FOL Problems

Undecidable

PRV, VAL, SAT
[Turing, 1936, Church, 1936]

Dyadic variant
[Kalmár, 1939]

Finite VAL, SAT
[Trakhtenbrot, 1950]

Decidable

Monadic variant
[Löwenheim, 1915]

Classifying FOL Problems

Undecidable	Decidable
PRV, VAL, SAT [Turing, 1936, Church, 1936]	
Dyadic variant [Kalmár, 1939]	**Monadic variant** [Löwenheim, 1915]
Finite VAL, SAT [Trakhtenbrot, 1950]	
Finite Dyadic variant	

Classifying FOL Problems

Undecidable	Decidable
$\mathsf{PRV}, \mathsf{VAL}, \mathsf{SAT}$ [Turing, 1936, Church, 1936]	
Dyadic variant [Kalmár, 1939]	**Monadic variant** [Löwenheim, 1915]
Finite $\mathsf{VAL}, \mathsf{SAT}$ [Trakhtenbrot, 1950]	
Finite Dyadic variant	**Finite Monadic variant**

Classifying FOL Problems

Undecidable	Decidable
For **full** logical fragment	
PRV, VAL, SAT [Turing, 1936, Church, 1936]	
Dyadic variant [Kalmár, 1939]	**Monadic variant** [Löwenheim, 1915]
Finite VAL, SAT [Trakhtenbrot, 1950]	
Finite Dyadic variant	**Finite Monadic variant**

Classifying FOL Problems

Undecidable	Decidable

For $(\forall, \rightarrow, \bot)$-fragment

Undecidable	Decidable
$\mathsf{PRV}, \mathsf{VAL}, \mathsf{SAT}$ [Turing, 1936, Church, 1936]	
Dyadic variant [Kalmár, 1939]	**Monadic variant** [Löwenheim, 1915]
Finite $\mathsf{VAL}, \mathsf{SAT}$ [Trakhtenbrot, 1950]	
Finite Dyadic variant	**Finite Monadic variant**

Classifying FOL Problems

Undecidable	Decidable

For $(\forall, \rightarrow)$-fragment

$\mathsf{PRV}, \mathsf{VAL}$	SAT
Dyadic variant [Kalmár, 1939]	**Monadic variant** [Löwenheim, 1915]
Finite VAL	**Finite** SAT
Finite Dyadic variant	**Finite Monadic variant**

Classifying FOL Problems

Undecidable	Decidable
Mechanizations	
PRV, VAL, SAT [Forster, Kirst, and Smolka, 2019]	
Dyadic variant [Kirst and Hermes, 2021]	**Monadic variant**
Finite VAL, SAT [Kirst and Larchey-Wendling, 2020]	
Finite Dyadic variant [Kirst and Larchey-Wendling, 2020]	**Finite Monadic variant**

Diophantine Constraints

- Hilberts 10th Problem: Algorithm for solving polynomial equation with integer coefficients

Diophantine Constraints

- Hilberts 10th Problem: Algorithm for solving polynomial equation with integer coefficients
- **DPRM Theorem**: No such algorithm exists [Matiyasevich, 1970]
- H10 is undecidable

Diophantine Constraints

- Hilberts 10th Problem: Algorithm for solving polynomial equation with integer coefficients
- **DPRM Theorem**: No such algorithm exists [Matiyasevich, 1970]
- H10 is undecidable

Diophantine Constraints: Variants of H10, like for example

- System of multiple polynomial equations with coefficients in $\mathbb{N}$
- System of equations, all of shape $a + b = c$, $a \cdot b = c$, or $a = 1$

Diophantine Constraints

- Hilberts 10th Problem: Algorithm for solving polynomial equation with integer coefficients
- **DPRM Theorem**: No such algorithm exists [Matiyasevich, 1970]
- H10 is undecidable

Diophantine Constraints: Variants of H10, like for example

- System of multiple polynomial equations with coefficients in $\mathbb{N}$
- System of equations, all of shape $a + b = c$, $a \cdot b = c$, or $a = 1$

Undecidability is mechanized in Coq [Larchey-Wendling and Forster, 2019]

Our Reductions

Our Reductions

We show:

- VAL, PRV undecidable for $(\forall, \rightarrow)$-fragment and **dyadic** signature
- $\overline{\text{SAT}}$, finite $\overline{\text{VAL}}$, finite SAT undecidable for $(\forall, \rightarrow, \bot)$-fragment and **dyadic** signature

Our Reductions

We show:

- VAL, PRV undecidable for $(\forall, \rightarrow)$-fragment and **dyadic** signature
- $\overline{\text{SAT}}$, finite $\overline{\text{VAL}}$, finite SAT undecidable for $(\forall, \rightarrow, \bot)$-fragment and **dyadic** signature
- **strongest** possible results (regarding signature and logical fragment)

Existing approaches: signature compression, classical syntax compression

Our Reductions

We show:

- VAL, PRV undecidable for $(\forall, \rightarrow)$-fragment and **dyadic** signature
- $\overline{\text{SAT}}$, finite $\overline{\text{VAL}}$, finite SAT undecidable for $(\forall, \rightarrow, \bot)$-fragment and **dyadic** signature
- **strongest** possible results (regarding signature and logical fragment)

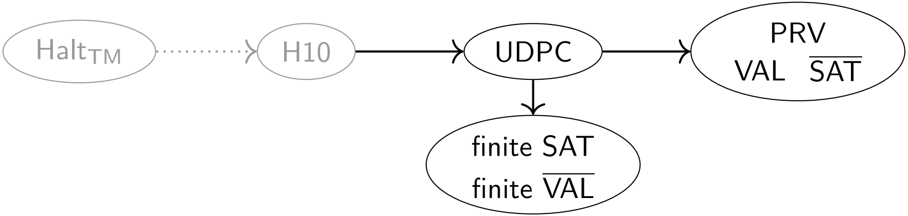
Existing approaches: signature compression, classical syntax compression
Our approach: **direct, compact** reductions
from UDPC, our variant of Diophantine constraints

Our Reductions

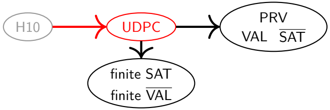
We show:

- VAL, PRV undecidable for $(\forall, \to)$-fragment and **dyadic** signature
- $\overline{\text{SAT}}$, finite $\overline{\text{VAL}}$, finite SAT undecidable for $(\forall, \to, \bot)$-fragment and **dyadic** signature
- **strongest** possible results (regarding signature and logical fragment)

Existing approaches: signature compression, classical syntax compression
Our approach: **direct, compact** reductions
from UDPC, our variant of Diophantine constraints



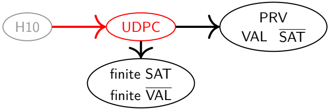
Uniform Diophantine Pair Constraints



$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := a + b + 1 = c \;\wedge\; d + d = b^2 + b$$

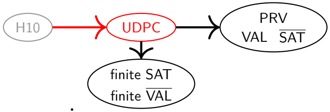
Uniform Diophantine Pair Constraints



$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \ \wedge \ d + d = b^2 + b$$

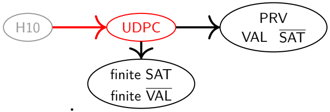
Uniform Diophantine Pair Constraints



$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \;\land\; \overbrace{d + d = b^2 + b}^{\text{Encodes squaring}}$$

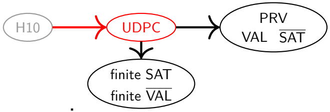
Uniform Diophantine Pair Constraints



$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \underbrace{\overbrace{d + d = b^2 + b}^{\text{Encodes squaring}}}_{\text{Gaussian sum}}$$

Uniform Diophantine Pair Constraints

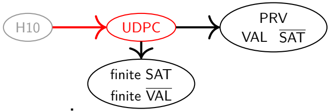


$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \;\wedge\; \underbrace{\overbrace{d + d = b^2 + b}^{\text{Encodes squaring}}}_{\text{Gaussian sum}}$$

$$(a, 0) \wr (a + 1, 0) := a + 1 = a + 1 \;\wedge\; 0 + 0 = 0^2 + 0$$

Uniform Diophantine Pair Constraints

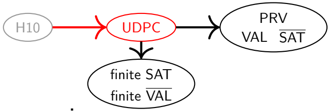


$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

Uniform Diophantine Pair Constraints



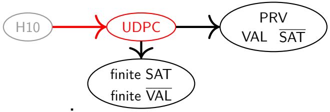
$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

Uniform Diophantine Pair Constraints



$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

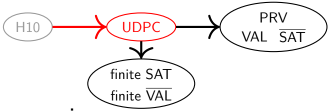
$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

- $\wr$ characterized as **inductive relation**
- $\wr$ can encode **any polynomial** on $\mathbb{N}$

Uniform Diophantine Pair Constraints



$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

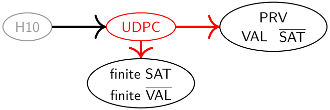
$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

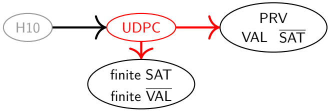
- $\wr$ characterized as **inductive relation**
- $\wr$ can encode **any polynomial** on $\mathbb{N}$
- UDPC: Given set of constraints of shape $\wr$, is there a solution?
  - Undecidable by reduction from Diophantine constraints

A first-order theory of ≀



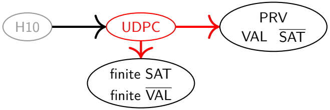
- Idea: Encode constructor laws

A first-order theory of $\wr$



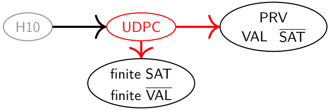
- Idea: Encode constructor laws
- Problem: $\wr$ is binary relation on pairs, but **characterized by their components**

A first-order theory of $\wr$



- Idea: Encode constructor laws
- Problem: $\wr$ is binary relation on pairs, but **characterized by their components**
- Solution: First-order theory on $\mathbb{N}$ and $\mathbb{N}^2$

A first-order theory of $\wr$



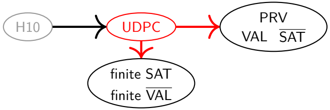
- Idea: Encode constructor laws
- Problem: $\wr$ is binary relation on pairs, but **characterized by their components**
- Solution: First-order theory on $\mathbb{N}$ and $\mathbb{N}^2$

  Standard model: $D := \mathbb{N} + \mathbb{N}^2$

  Interpretation of $\wr\wr$:

$l \setminus r$	$y : \mathbb{N}$	$(c, d) : \mathbb{N}^2$
$x : \mathbb{N}$	$x = y$	$x = c$
$(a, b) : \mathbb{N}^2$	$y = b$	$(a, b) \wr (c, d)$

A first-order theory of $\wr$



- Idea: Encode constructor laws
- Problem: $\wr$ is binary relation on pairs, but **characterized by their components**
- Solution: First-order theory on $\mathbb{N}$ and $\mathbb{N}^2$

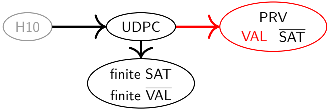
  Standard model: $D := \mathbb{N} + \mathbb{N}^2$

  Interpretation of $\wr\wr$:

l \ r	$y : \mathbb{N}$	$(c, d) : \mathbb{N}^2$
$x : \mathbb{N}$	$x = y$	$x = c$
$(a, b) : \mathbb{N}^2$	$y = b$	$(a, b) \wr (c, d)$

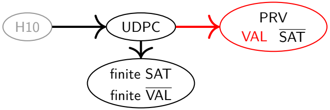
- Build formula $\varphi_{\wr}(a, b, c, d)$ encoding $(a, b) \wr (c, d)$

Reducing to VAL



Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

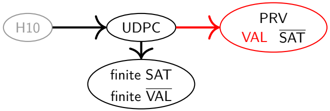
Reducing to VAL



Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

Reducing to VAL

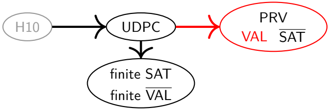


Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$[(a, b) \wr (a, a),\ (b, c) \wr (b, b)]$$

Reducing to VAL

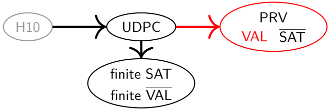


Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$\varphi_\wr(a, b, a, a) \land \varphi_\wr(b, c, b, b)$$

Reducing to VAL

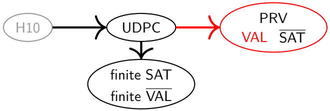


Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$\exists abc : \varphi_{\wr}(a, b, a, a) \land \varphi_{\wr}(b, c, b, b)$$

Reducing to VAL

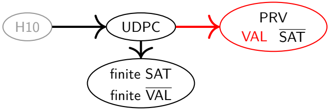


Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$Ax_1 \to Ax_2 \to Ax_3 \to \exists abc : \varphi_\wr(a, b, a, a) \land \varphi_\wr(b, c, b, b)$$

Reducing to VAL



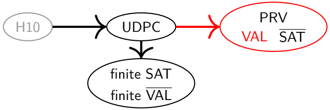
Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$Ax_1 \rightarrow Ax_2 \rightarrow Ax_3 \rightarrow \exists abc : \varphi_\wr(a, b, a, a) \wedge \varphi_\wr(b, c, b, b)$$

- F is computable since it is implemented in Coq

Reducing to VAL



Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

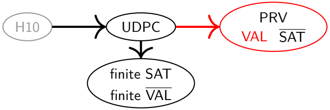
1. Find $F(h)$:

$$Ax_1 \to Ax_2 \to Ax_3 \to \exists abc : \varphi_\wr(a, b, a, a) \land \varphi_\wr(b, c, b, b)$$

   - F is computable since it is implemented in Coq

2. VAL $F(h) \Rightarrow$ UDPC h
   - **Extract** using standard model
   - Axioms are satisfied by standard model

Reducing to VAL



Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

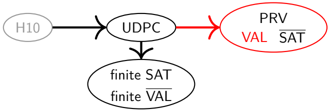
1. Find $F(h)$:

$$Ax_1 \to Ax_2 \to Ax_3 \to \exists abc : \varphi_{\wr}(a, b, a, a) \wedge \varphi_{\wr}(b, c, b, b)$$

   - F is computable since it is implemented in Coq

2. VAL $F(h) \Rightarrow$ UDPC h
   - **Extract** using standard model
   - Axioms are satisfied by standard model ✓

Reducing to VAL



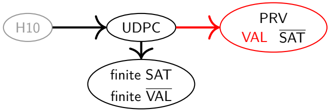
Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$Ax_1 \rightarrow Ax_2 \rightarrow Ax_3 \rightarrow \exists abc : \varphi_\wr(a, b, a, a) \land \varphi_\wr(b, c, b, b)$$

   - F is computable since it is implemented in Coq
2. VAL $F(h) \Rightarrow$ UDPC h
   - **Extract** using standard model
   - Axioms are satisfied by standard model ✓
3. UDPC $h \Rightarrow$ VAL $F(h)$
   - **Encode** proof of $(a, b) \wr (c, d)$ into model $\mathcal{M}$

Reducing to VAL



Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

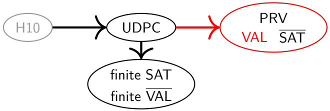
$$Ax_1 \to Ax_2 \to Ax_3 \to \exists abc : \varphi_\wr(a, b, a, a) \land \varphi_\wr(b, c, b, b)$$

   - F is computable since it is implemented in Coq

2. VAL $F(h) \Rightarrow$ UDPC h
   - **Extract** using standard model
   - Axioms are satisfied by standard model ✓

3. UDPC $h \Rightarrow$ VAL $F(h)$
   - **Encode** proof of $(a, b) \wr (c, d)$ into model $\mathcal{M}$
   - Induction on $\wr$

Reducing to VAL



Given a collection of constraints h, create formula $F(h)$ valid iff h has solution.

1. Find $F(h)$:

$$Ax_1 \to Ax_2 \to Ax_3 \to \exists abc : \varphi_\wr(a, b, a, a) \land \varphi_\wr(b, c, b, b)$$

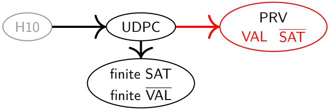
   - F is computable since it is implemented in Coq

2. VAL $F(h) \Rightarrow$ UDPC h
   - **Extract** using standard model
   - Axioms are satisfied by standard model ✓

3. UDPC $h \Rightarrow$ VAL $F(h)$
   - **Encode** proof of $(a, b) \wr (c, d)$ into model $\mathcal{M}$
   - Induction on $\wr$
   - First find initial fragment of $\mathbb{N}$ in $\mathcal{M}$

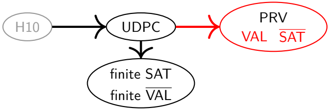
$\Rightarrow$ VAL **undecidable** for dyadic signature

Sharper results for $\mathsf{VAL}, \mathsf{SAT}$



Restrict the admissible logical operators to $\forall, \rightarrow$

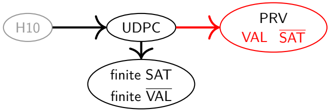
Sharper results for VAL, SAT



Restrict the admissible logical operators to $\forall, \rightarrow$

1. **Negative translation** [Gödel, 1933, Gentzen, 1936]
   - Replace $\exists$ by $\neg\forall\neg$ etc.
   - Does not work in intuitionistic logic

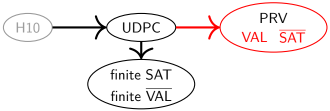
Sharper results for VAL, SAT



Restrict the admissible logical operators to $\forall, \rightarrow$

1. **Negative translation** [Gödel, 1933, Gentzen, 1936]
   - Replace $\exists$ by $\neg\forall\neg$ etc.
   - Does not work in intuitionistic logic
2. **Friedman's A-translation** [Friedman, 1978]
   - Replace $\bot$ by some formula A...
   - such that standard interpretation of A is UDPC h...
   - without introducing additional relation symbols

Sharper results for VAL, SAT

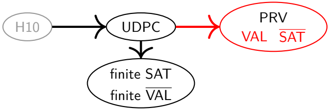


Restrict the admissible logical operators to $\forall, \rightarrow$

1. **Negative translation** [Gödel, 1933, Gentzen, 1936]
   - Replace $\exists$ by $\neg\forall\neg$ etc.
   - Does not work in intuitionistic logic
2. **Friedman's A-translation** [Friedman, 1978]
   - Replace $\bot$ by some formula A...
   - such that standard interpretation of A is UDPC h...
   - <u>without</u> introducing additional relation symbols

- Combine both to preserve validity
   - By cleverly choosing A, we can escape double negation once

Sharper results for VAL, SAT



Restrict the admissible logical operators to $\forall, \rightarrow$

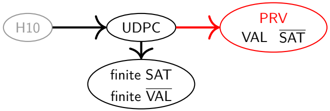
1. **Negative translation** [Gödel, 1933, Gentzen, 1936]
   - Replace $\exists$ by $\neg\forall\neg$ etc.
   - Does not work in intuitionistic logic
2. **Friedman's A-translation** [Friedman, 1978]
   - Replace $\bot$ by some formula A...
   - such that standard interpretation of A is UDPC h...
   - <u>without</u> introducing additional relation symbols

- Combine both to preserve validity
  - By cleverly choosing A, we can escape double negation once

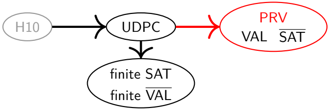
In summary:

- VAL **undecidable** for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT **undecidable** for dyadic signature $(\forall, \rightarrow, \bot)$-fragment

Provability

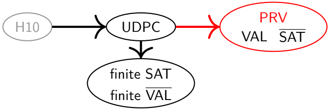


Provability



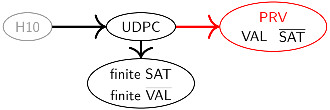
- In classical meta-theory: $\text{VAL}\,\varphi \Rightarrow \text{PRV}_c\,\varphi$ [Gödel, 1930]

Provability



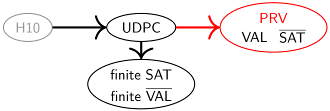
- In classical meta-theory: $\text{VAL}\,\varphi \Rightarrow \text{PRV}_c\,\varphi$ [Gödel, 1930]
- In intuitionistic meta-theory: $\text{VAL}\,\varphi \Rightarrow \neg\neg\text{PRV}_c\,\varphi$ for $(\forall, \rightarrow, \bot)$-fragment

Provability



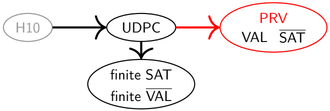
- In classical meta-theory: $\text{VAL}\,\varphi \Rightarrow \text{PRV}_c\,\varphi$ [Gödel, 1930]
- In intuitionistic meta-theory: $\text{VAL}\,\varphi \Rightarrow \neg\neg\text{PRV}_c\,\varphi$ for $(\forall, \rightarrow, \bot)$-fragment
- **Soundness** $\text{PRV}_i\,\varphi \Rightarrow \text{VAL}\,\varphi$ still holds

Provability



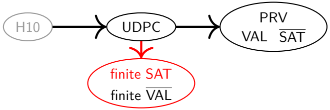
- In classical meta-theory: $\text{VAL}\,\varphi \Rightarrow \text{PRV}_c\,\varphi$ [Gödel, 1930]
- In intuitionistic meta-theory: $\text{VAL}\,\varphi \Rightarrow \neg\neg\text{PRV}_c\,\varphi$ for $(\forall, \rightarrow, \bot)$-fragment
- **Soundness** $\text{PRV}_i\,\varphi \Rightarrow \text{VAL}\,\varphi$ still holds
- Dedicated reduction to (intuitionistic) PRV using **same reduction function**
- Results:
  - PRV **undecidable** for dyadic signature over $(\forall, \rightarrow)$-fragment
  - Kripke semantics **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

Provability



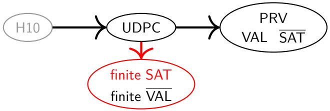
- In classical meta-theory: $\text{VAL}\,\varphi \Rightarrow \text{PRV}_c\,\varphi$ [Gödel, 1930]
- In intuitionistic meta-theory: $\text{VAL}\,\varphi \Rightarrow \neg\neg\text{PRV}_c\,\varphi$ for $(\forall, \rightarrow, \bot)$-fragment
- **Soundness** $\text{PRV}_i\,\varphi \Rightarrow \text{VAL}\,\varphi$ still holds
- Dedicated reduction to (intuitionistic) PRV using **same reduction function**
- Results:
  - PRV **undecidable** for dyadic signature over $(\forall, \rightarrow)$-fragment
  - Kripke semantics **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
  - Classical provability PRV_c **similarly undecidable**, assuming LEM

Finite Satisfiability



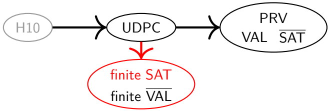
- Restrict models to finite types
  - Follow existing mechanization [Kirst and Larchey-Wendling, 2020]
  - Finite models also have **decidable predicates**

Finite Satisfiability



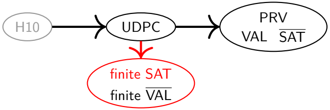
- Restrict models to finite types
  - Follow existing mechanization [Kirst and Larchey-Wendling, 2020]
  - Finite models also have **decidable predicates**
- finite SAT mechanization build **inversely** to previous reduction
  - Before: Encode given solution into model
  - Now: Given model encoding solution, **extract** it

Finite Satisfiability



- Restrict models to finite types
  - Follow existing mechanization [Kirst and Larchey-Wendling, 2020]
  - Finite models also have **decidable predicates**
- finite SAT mechanization build **inversely** to previous reduction
  - Before: Encode given solution into model
  - Now: Given model encoding solution, **extract** it
  - Axioms resembling **eliminators**
- Finite standard model: $M = \mathbb{N}_{\leq m} + \mathbb{N}^2_{\leq m}$

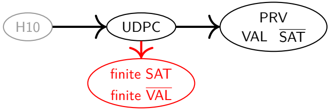
Finite Satisfiability



- Restrict models to finite types
  - Follow existing mechanization [Kirst and Larchey-Wendling, 2020]
  - Finite models also have **decidable predicates**
- finite SAT mechanization build **inversely** to previous reduction
  - Before: Encode given solution into model
  - Now: Given model encoding solution, **extract** it
  - Axioms resembling **eliminators**
- Finite standard model: $M = \mathbb{N}_{\leq m} + \mathbb{N}^2_{\leq m}$

$\Rightarrow$ Finite SAT is **undecidable** for dyadic signature

Finite Satisfiability



- Restrict models to finite types
  - Follow existing mechanization [Kirst and Larchey-Wendling, 2020]
  - Finite models also have **decidable predicates**
- finite SAT mechanization build **inversely** to previous reduction
  - Before: Encode given solution into model
  - Now: Given model encoding solution, **extract** it
  - Axioms resembling **eliminators**
- Finite standard model: $M = \mathbb{N}_{\leq m} + \mathbb{N}^2_{\leq m}$

$\Rightarrow$ Finite SAT is **undecidable** for dyadic signature

$\Rightarrow$ Finite $\overline{\text{VAL}}$ is **undecidable** for dyadic signature

$\Rightarrow$ Negative translation yields same results for $(\forall, \rightarrow, \bot)$-fragment

Analysis and Comparison

Working in object logics

We use existing mechanization of first-order logic [Kirst et al., 2022]

- Formulated using de Bruijn binders
  - Nice for meta-theory (e.g. deductive Weakening lemma)
  - Bad user experience when used as input language
  - Working within concrete model is intricate

Working in object logics

We use existing mechanization of first-order logic [Kirst et al., 2022]

- Formulated using de Bruijn binders
  - Nice for meta-theory (e.g. deductive Weakening lemma)
  - Bad user experience when used as input language
  - Working within concrete model is intricate
- Syntactic proofs are cumbersome to construct
  - Lots of reasoning for trivial properties
  - No automation
  - Managing the context again requires analyzing de Bruijn binders

Working in object logics

We use existing mechanization of first-order logic [Kirst et al., 2022]

- Formulated using de Bruijn binders
  - Nice for meta-theory (e.g. deductive Weakening lemma)
  - Bad user experience when used as input language
  - Working within concrete model is intricate
- Syntactic proofs are cumbersome to construct
  - Lots of reasoning for trivial properties
  - No automation
  - Managing the context again requires analyzing de Bruijn binders
- Idea: Develop toolbox easing these tasks [Hostert et al., 2021]

Comparison of FOL undecidability/reductions

Paper	Dyadic signature	Small fragment	Coq	Reduction
[Church, 1936]	$\times$	$\times$	$\times$	λ-calculus
[Turing, 1936]	$\times$	$\times$	$\times$	Turing machines

Comparison of FOL undecidability/reductions

Paper	Dyadic signature	Small fragment	Coq	Reduction
[Church, 1936]	$\times$	$\times$	$\times$	λ-calculus
[Turing, 1936]	$\times$	$\times$	$\times$	Turing machines
[Kalmár, 1937]	$\checkmark$	$\times$	$\times$	signature compression
[Gentzen, 1936]	$\checkmark$	$\checkmark$[1]	$\times$	negative translation

[1] $(\forall, \rightarrow, \wedge, \bot)$

Comparison of FOL undecidability/reductions

Paper	Dyadic signature	Small fragment	Coq	Reduction
[Church, 1936]	×	×	×	λ-calculus
[Turing, 1936]	×	×	×	Turing machines
[Kalmár, 1937]	✓	×	×	signature compression
[Gentzen, 1936]	✓	✓[1]	×	negative translation
[Forster, Kirst, and Smolka, 2019]	×	✓	✓	PCP
[Kirst and Hermes, 2021]	✓	×	✓	PCP via ZF

[1] $(\forall, \rightarrow, \wedge, \bot)$

Comparison of FOL undecidability/reductions

Paper	Dyadic signature	Small fragment	Coq	Reduction
[Church, 1936]	×	×	×	λ-calculus
[Turing, 1936]	×	×	×	Turing machines
[Kalmár, 1937]	✓	×	×	signature compression
[Gentzen, 1936]	✓	✓[1]	×	negative translation
[Forster, Kirst, and Smolka, 2019]	×	✓	✓	PCP
[Kirst and Hermes, 2021]	✓	×	✓	PCP via ZF
The present work	✓	✓	✓	UDPC/H10

[1] $(\forall, \rightarrow, \wedge, \bot)$

Comparison of Finite Satisfiability results

Paper	Dyadic signature	Small fragment	Coq	Method
[Trakhtenbrot, 1950]	$\times$	$\times$	$\times$	μ-recursive functions

Comparison of Finite Satisfiability results

Paper	Dyadic signature	Small fragment	Coq	Method
[Trakhtenbrot, 1950]	×	×	×	μ-recursive functions
[Kalmár, 1937]	?	×	×	signature compression

Comparison of Finite Satisfiability results

Paper	Dyadic signature	Small fragment	Coq	Method
[Trakhtenbrot, 1950]	$\times$	$\times$	$\times$	μ-recursive functions
[Kalmár, 1937]	?	$\times$	$\times$	signature compression
[Libkin, 2004]	($\checkmark$)	$\times$	$\times$	TM

Comparison of Finite Satisfiability results

Paper	Dyadic signature	Small fragment	Coq	Method
[Trakhtenbrot, 1950]	$\times$	$\times$	$\times$	μ-recursive functions
[Kalmár, 1937]	?	$\times$	$\times$	signature compression
[Libkin, 2004]	($\checkmark$)	$\times$	$\times$	TM
[Kirst and Larchey-Wendling, 2020]	$\checkmark$	$\times$	$\checkmark$	PCP signature compression

Comparison of Finite Satisfiability results

Paper	Dyadic signature	Small fragment	Coq	Method
[Trakhtenbrot, 1950]	×	×	×	μ-recursive functions
[Kalmár, 1937]	?	×	×	signature compression
[Libkin, 2004]	(✓)	×	×	TM
[Kirst and Larchey-Wendling, 2020]	✓	×	✓	PCP signature compression
The present work	✓	✓	✓	UDPC/H10

Summary

We contribute the mechanized undecidability of

- PRV, VAL for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT, finite SAT, finite VAL for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

Summary

We contribute the mechanized undecidability of

- PRV, VAL for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT, finite SAT, finite VAL for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Using UDPC, a **novel** decision problem suitable for **compact, direct reductions**

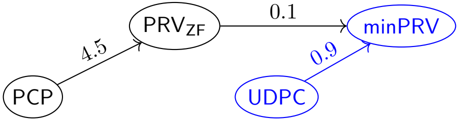
Summary

We contribute the mechanized undecidability of

- PRV, VAL for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT, finite SAT, finite VAL for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Using UDPC, a **novel** decision problem suitable for **compact, direct reductions**

Coq mechanization:

- ~900 LoC for PRV and corollaries
  - [Kirst and Hermes, 2021]: 4.5k LoC



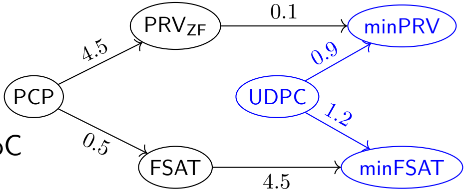
Summary

We contribute the mechanized undecidability of

- PRV, VAL for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT, finite SAT, finite VAL for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Using UDPC, a **novel** decision problem suitable for **compact, direct reductions**

Coq mechanization:

- ~900 LoC for PRV and corollaries
- ~1200 LoC for finite SAT, VAL
  - [Kirst and Larchey-Wendling, 2020]: >5k LoC



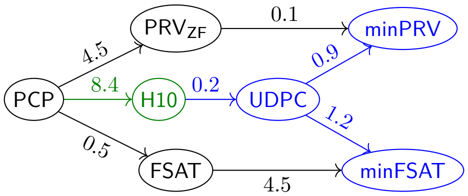
Summary

We contribute the mechanized undecidability of

- PRV, VAL for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT, finite SAT, finite VAL for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Using UDPC, a **novel** decision problem suitable for **compact, direct reductions**

Coq mechanization:

- ~900 LoC for PRV and corollaries
- ~1200 LoC for finite SAT, VAL
- ~200 LoC for UDPC
- Requires undecidability of H10:
  8k LoC [Larchey-Wendling and Forster, 2019]



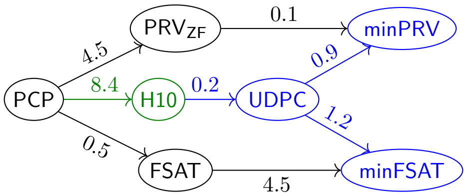
Summary

We contribute the mechanized undecidability of

- PRV, VAL for dyadic signature over $(\forall, \rightarrow)$-fragment
- SAT, finite SAT, finite VAL for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Using UDPC, a **novel** decision problem suitable for **compact, direct reductions**

Coq mechanization:

- ~900 LoC for PRV and corollaries
- ~1200 LoC for finite SAT, VAL
- ~200 LoC for UDPC
- Requires undecidability of H10:
  8k LoC [Larchey-Wendling and Forster, 2019]



Work contributed to the Coq Library of Undecidability Proofs [Forster et al., 2020]

`https://www.ps.uni-saarland.de/extras/fol-dyadic/`

Bibliography I

[Bauer, 2006] Bauer, A. (2006). First Steps in Synthetic Computability Theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31. Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI).

[Church, 1936] Church, A. (1936). A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):4041.

[Forster, 2021] Forster, Y. (2021). *Computability in Constructive Type Theory*. PhD thesis, Saarland University.

[Forster et al., 2020] Forster, Y., Larchey-Wendling, D., Dudenhefner, A., Heiter, E., Kirst, D., Kunze, F., Smolka, G., Spies, S., Wehr, D., and Wuttke, M. (2020). A Coq library of undecidable problems. In *CoqPL 2020 The Sixth International Workshop on Coq for Programming Languages*.

[Forster, Kirst, and Smolka, 2019] Forster, Kirst, and Smolka (2019). On Synthetic Undecidability in Coq, with an Application to the Entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2019, page 3851, New York, NY, USA. Association for Computing Machinery.

[Friedman, 1978] Friedman, H. (1978). Classically and intuitionistically provably recursive functions. In Müller, G. H. and Scott, D. S., editors, *Higher Set Theory*, pages 21–27, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Gentzen, 1936] Gentzen, G. (1936). Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112:493–565.

Bibliography II

[Gödel, 1933] Gödel, K. (1928-1933). Zur intuitionistischen Arithmetik und Zahlentheorie – Ergebnisse eines Mathematischen Kolloquiums. pages 493–565.

[Gödel, 1930] Gödel, K. (1930). Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37(1):349–360.

[Hostert et al., 2021] Hostert, J., Koch, M., and Kirst, D. (2021). A Toolbox for Mechanised First-Order Logic. In *The Coq Workshop 2021*.

[Kalmár, 1937] Kalmár, L. (1937). Zurückführung des Entscheidungsproblems auf den Fall von Formeln mit einer einzigen, binären, Funktionsvariablen. *Compositio Mathematica*, 4:137–144.

[Kalmár, 1939] Kalmár, L. (1939). On the reduction of the decision problem. First paper. Ackermann prefix, a single binary predicate. *Journal of Symbolic Logic*, 4(1):19.

[Kirst and Hermes, 2021] Kirst, D. and Hermes, M. (2021). Synthetic Undecidability and Incompleteness of First-Order Axiom Systems in Coq. In *Interactive Theorem Proving - 12th International Conference, ITP 2021, Rome, Italy*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[Kirst et al., 2022] Kirst, D., Hostert, J., Dudenhefner, A., Forster, Y., Hermes, M., Koch, M., Larchey-Wendling, D., Mück, N., Peters, B., Smolka, G., and Wehr, D. (2022). A Coq Library for Mechanised First-Order Logic. In *The Coq Workshop 2022*.

Bibliography III

[Kirst and Larchey-Wendling, 2020] Kirst, D. and Larchey-Wendling, D. (2020). Trakhtenbrot's Theorem in Coq. In Peltier, N. and Sofronie-Stokkermans, V., editors, *Automated Reasoning*, pages 79–96, Cham. Springer International Publishing.

[Larchey-Wendling and Forster, 2019] Larchey-Wendling, D. and Forster, Y. (2019). Hilbert's Tenth Problem in Coq. 4th International Conference on Formal Structures for Computation and Deduction.

[Libkin, 2004] Libkin, L. (2004). *Elements of Finite Model Theory*. Springer.

[Löwenheim, 1915] Löwenheim, L. (1915). Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, 76:447–470.

[Matiyasevich, 1970] Matiyasevich, Y. V. (1970). Enumerable sets are Diophantine. *Doklady Akademii Nauk SSSR*, 191:279–282.

[Richman, 1983] Richman, F. (1983). Church's thesis without tears. *Journal of Symbolic Logic*, 48(3):797803.

[Trakhtenbrot, 1950] Trakhtenbrot, B. (1950). The Impossibility of an Algorithm for the Decidability Problem on Finite Classes. In *Proceedings of the USSR Academy of Sciences*.

[Turing, 1936] Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265.

Synthetic Undecidability

Classical/Textbook approach:

Synthetic Undecidability

Classical/Textbook approach:

Model of computation

Synthetic Undecidability

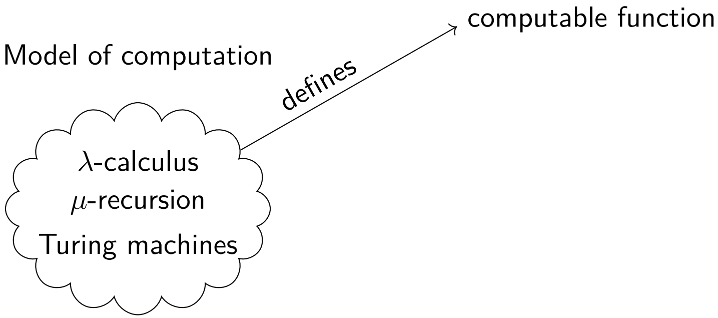
Classical/Textbook approach:

Model of computation

λ-calculus
μ-recursion
Turing machines

Synthetic Undecidability

Classical/Textbook approach:



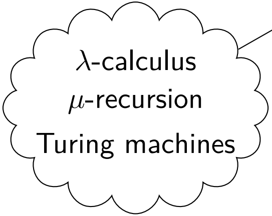
Synthetic Undecidability

Classical/Textbook approach:

computable function

Model of computation

defines



defines

predicate P decidable:

$$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\, x \Leftrightarrow f\, x = \texttt{tt})$$
$$\wedge \text{ computable } f$$

Synthetic Undecidability

Classical/Textbook approach:

Model of computation

- λ-calculus
- μ-recursion
- Turing machines

defines → computable function

computable function defines → reduction $P \preceq Q$:

$$\exists F.\, (\forall x.\, P\,x \Leftrightarrow Q(F\,x))$$
$$\land \text{ computable } F$$

computable function defines → predicate P decidable:

$$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\,x \Leftrightarrow f\,x = \texttt{tt})$$
$$\land \text{ computable } f$$

Synthetic Undecidability

Classical/Textbook approach:

Model of computation

- λ-calculus
- μ-recursion
- Turing machines

defines → computable function

computable function — defines → reduction $P \preceq Q$:

$$\exists F.\, (\forall x.\, P\,x \Leftrightarrow Q(F\,x))$$
$$\land \text{ computable } F$$

computable function — defines → predicate P decidable:

$$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\,x \Leftrightarrow f\,x = \texttt{tt})$$
$$\land \text{ computable } f$$

P undecidable:
$\neg(\text{decidable } P)$

Synthetic Undecidability

Classical/Textbook approach:

Model of computation

- λ-calculus
- μ-recursion
- Turing machines

Model of computation —defines→ computable function

computable function —defines→ reduction $P \preceq Q$:

$$\exists F.\, (\forall x.\, P\, x \Leftrightarrow Q(F\, x))$$

$\land$ computable F

computable function —defines→ predicate P decidable:

$$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\, x \Leftrightarrow f\, x = \texttt{tt})$$

$\land$ computable f

P undecidable:
$\neg$(decidable P)

Synthetic Undecidability

Classical/Textbook approach:

computable function

Model of computation

λ-calculus
μ-recursion
Turing machines

defines

defines

defines

reduction $P \preceq Q$:

$\exists F.\, (\forall x.\, P\,x \Leftrightarrow Q(F\,x))$

$\wedge$ computable F

predicate P decidable:

$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\,x \Leftrightarrow f\,x = \texttt{tt})$

$\wedge$ computable f

P undecidable:

$\neg$(decidable P)

Synthetic Undecidability

Synthetic approach:

computable function

defines

defines

predicate P decidable:

$$\exists f : \mathbb{N} \to \mathbb{B}. \, (\forall x. \, P\,x \Leftrightarrow f\,x = \texttt{tt})$$

$\wedge$ computable f

reduction $P \preceq Q$:

$$\exists F. \, (\forall x. \, P\,x \Leftrightarrow Q(F\,x))$$

$\wedge$ computable F

P undecidable:

$\neg$(decidable P)

Synthetic Undecidability

Synthetic approach:

All functions defined in constructive type theory are computable

computable function

defines

defines

predicate P decidable:

$\exists f : \mathbb{N} \to \mathbb{B}. (\forall x. P\,x \Leftrightarrow f\,x = \texttt{tt})$

$\wedge$ computable f

reduction $P \preceq Q$:

$\exists F. (\forall x. P\,x \Leftrightarrow Q(F\,x))$

$\wedge$ computable F

P undecidable:

$\neg$(decidable P)

Synthetic Undecidability

Synthetic approach:

All functions defined
in constructive type theory
are computable

Any function is a
computable function

defines

defines

predicate P decidable:

$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\, x \Leftrightarrow f\, x = \texttt{tt})$

$\wedge$ computable f

reduction $P \preceq Q$:

$\exists F.\, (\forall x.\, P\, x \Leftrightarrow Q(F\, x))$

$\wedge$ computable F

P undecidable:
$\neg$(decidable P)

Synthetic Undecidability

Synthetic approach:

All functions defined
in constructive type theory
are computable

Any function is a
computable function

defines

defines

predicate P decidable:

$\exists f : \mathbb{N} \to \mathbb{B}.\, (\forall x.\, P\, x \Leftrightarrow f\, x = \texttt{tt})$

reduction $P \preceq Q$:

$\exists F.\, (\forall x.\, P\, x \Leftrightarrow Q(F\, x))$

P undecidable:
$\neg$(decidable P)

Synthetic Undecidability

Synthetic approach:

All functions defined
in constructive type theory
are computable

Any function is a
computable function

defines

defines

predicate P decidable:

$\exists f : \mathbb{N} \to \mathbb{B}. (\forall x. P\, x \Leftrightarrow f\, x = \texttt{tt})$

reduction $P \preceq Q$:

$\exists F. (\forall x. P\, x \Leftrightarrow Q(F\, x))$

P undecidable:
decidable $P \Rightarrow$ decidable $HALT_{\mathsf{TM}}$

Synthetic Undecidability

Synthetic approach:

All functions defined
in constructive type theory
are computable

Any function is a
computable function

defines

defines

predicate P decidable:

$\exists f : \mathbb{N} \to \mathbb{B}. (\forall x. P\,x \Leftrightarrow f\,x = \texttt{tt})$

reduction $P \preceq Q$:

$\exists F. (\forall x. P\,x \Leftrightarrow Q(F\,x))$

P undecidable:
decidable $P \Rightarrow$ enumerable $\overline{HALT_{\mathsf{TM}}}$

Deduction system

C $\dfrac{\Gamma \vdash \varphi}{\varphi \in \Gamma}$ E $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \bot}$ II $\dfrac{\Gamma \vdash \varphi \to \psi}{\Gamma, \varphi \vdash \psi}$ IE $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}$

CI $\dfrac{\Gamma \vdash \varphi \wedge \psi}{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}$ CE_1 $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \wedge \psi}$ CE_2 $\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}$

DI_1 $\dfrac{\Gamma \vdash \varphi \vee \psi}{\Gamma \vdash \varphi}$ DI_2 $\dfrac{\Gamma \vdash \varphi \vee \psi}{\Gamma \vdash \psi}$ DE $\dfrac{\Gamma \vdash \theta}{\Gamma \vdash \varphi \vee \psi \qquad \Gamma, \varphi \vdash \theta \qquad \Gamma, \psi \vdash \theta}$

AI $\dfrac{\Gamma \vdash \forall \varphi}{\Gamma[\uparrow] \vdash \varphi}$ AE $\dfrac{\Gamma \vdash \varphi[t]}{\Gamma \vdash \forall \varphi}$ EI $\dfrac{\Gamma \vdash \exists \varphi}{\Gamma \vdash \varphi[t]}$ EE $\dfrac{\Gamma \vdash \psi}{\Gamma \vdash \exists \varphi \qquad \Gamma[\uparrow], \varphi \vdash \psi[\uparrow]}$

$\Gamma \vdash_c \varphi$ also has Pierce rule $((\varphi \to \psi) \to \varphi) \to \varphi$.

[Trakhtenbrot, 1950]

- Very ancient notation
- Given a general-recursive function f, construct formula $\mathfrak{U}$ that is finitely satisfied only if f has a root
- Construction by induction on syntax of f
- Paper leaves actual construction to the reader
- Reduction is an interesting approach which might be elegantly mechanizable
- Paper is not concerned with minimal representation
  - [Kalmár, 1937] already published a reduction from FOL to FOL with minimal signature
  - [Kalmár, 1937] claims the reduction should work for finite models without presenting proof
  - The fact that one can reduce to a dyadic signature was folklore knowledge in 1950

[Kirst and Larchey-Wendling, 2020]

Part on Trakhenbrot:

- Show *FSAT* undecidable by reducing from *PCP*
- Signature compression chain:
  - Arbitrary FOL with equality to arbitrary FOL without equality
    - Take quotient over first-order indistinguishability
  - Arbitrary FOL to single predicate FOL
    - Actually three different reductions
    - Compress functions to predicates
    - Compress predicates to one predicate $+$ unary functions
    - Compress functions to free variables
  - single predicate to dyadic predicate
    - Construction using $\in$ and HF-sets

Other results:

- Monadic signature is shown decidable
  - Function and relation symbols have arity ≤ 1, or
  - Relation symbols have arity 0

[Libkin, 2004]

- Textbook on Finite Model Theory
- Interesting section for us is 9.1
- Reduction from Turing Machine Halting Problem to *FSAT*
- Making this use minimal signature is (explicitly) left to the reader

The full reduction

1. Syntactic sugar:
   - $N\,k := k\#k$
   - $P'\,k := k\#k \to \bot$
   - $P\,p\,l\,r := P'\,p \land N\,l \land N\,r \land l\#p \land p\#r$
   - $(a, b)\#(c, d) := \exists p\,q, P\,p\,a\,b \land P\,q\,c\,d \land p\#q$
   - $x \equiv y := \forall k, k\#x \leftrightarrow k\#y \land x\#k \leftrightarrow y\#k$
   - $x \leq y := N\,x \land N\,y \land x\#y$
   - $x < y := x \leq y \land x \not\equiv y$
   - $rel\,a\,b\,c\,d\,m := (a, b)\#(c, d) \land a \leq m \land b \leq m \land c \leq m \land d \leq m$
2. Axioms:
   - $\forall xyz, x < y \to y < z \to x < z$
   - $\forall a, N\,a \to a \not\equiv 0 \to \exists a', (a', 0)\#(a, 0)$
   - $\forall ab, (a, 0)\#(b, 0) \to a < b \land \forall k, k < b \to k \leq a$
   - $\forall abcd, (a, b)\#(c, d) \to b \not\equiv 0 \to$
     $\exists b'c'd', (b', 0)\#(b, 0) \land (c', 0)\#(c, 0) \land (a, b')\#(c', d') \land (d', b')\#(d, d') \land d' < d$
   - $\forall acd, (a, 0)\#(c, d) \to d \equiv 0$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := a + b + 1 = c \land d + d = b^2 + b$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land d + d = b^2 + b$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{d + d = b^2 + b}^{\text{Encodes squaring}}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$(a, 0) \wr (a + 1, 0) := a + 1 = a + 1 \land 0 + 0 = 0^2 + 0$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$(a, 0) \wr (a + 1, 0)$ is an axiom

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b' + 1) \wr (c, d) = a + (b' + 1) + 1 = c \land 2 \cdot d = (b' + 1)^2 + b' + 1$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b' + 1) \wr (c, d) = a + (b' + 1) + 1 = c \wedge 2 \cdot d = b'^2 + b' + 2b' + 2$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b' + 1) \wr (c, d) = a + (b' + 1) + 1 = c \wedge 2 \cdot d = 2 \cdot d' + 2b' + 2$$

where $2 \cdot d' = b'^2 + b'$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b' + 1) \wr (c, d) = a + (b' + 1) + 1 = c \land d = d' + b' + 1$$

where $2 \cdot d' = b'^2 + b'$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b' + 1) \wr (c, d) = a + b' + 1 = c' \wedge d = d' + b' + 1$$

where $2 \cdot d' = b'^2 + b'$

and $c = c' + 1$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b) \wr (c, d) = a + b' + 1 = c' \land d = d' + b' + 1$$

where $2 \cdot d' = b'^2 + b'$

and $c = c' + 1$

and $b = b' + 1$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b) \wr (c, d) = a + b' + 1 = c' \wedge d = d' + b' + 1$$

where $(?, b') \wr (?, d')$

and $(c', 0) \wr (c, 0)$

and $(b', 0) \wr (b, 0)$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$(a, b) \wr (c, d) \;\Leftarrow\; a + b' + 1 = c' \land d = d' + b' + 1$$

and $(?, b') \wr (?, d')$

and $(c', 0) \wr (c, 0)$

and $(b', 0) \wr (b, 0)$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\begin{aligned}(a, b) \wr (c, d) \;\Leftarrow\;\; & d = d' + b' + 1 \\ & \text{and } (a, b') \wr (c', d') \\ & \text{and } (c', 0) \wr (c, 0) \\ & \text{and } (b', 0) \wr (b, 0)\end{aligned}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\begin{aligned} (a, b) \wr (c, d) \ \Leftarrow\ & (d', b') \wr (d, ?) \\ \text{and } & (a, b') \wr (c', d') \\ \text{and } & (c', 0) \wr (c, 0) \\ \text{and } & (b', 0) \wr (b, 0) \end{aligned}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \land \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\begin{aligned} (a, b) \wr (c, d) \ \Leftarrow\ & (d', b') \wr (d, d') \\ \text{and } & (a, b') \wr (c', d') \\ \text{and } & (c', 0) \wr (c, 0) \\ \text{and } & (b', 0) \wr (b, 0) \end{aligned}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

- $\wr$ as inductive relation axiomatizes itself

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

- $\wr$ as inductive relation axiomatizes itself
- $\wr$ can encode any equation on $\mathbb{N}$

Uniform Diophantine Pair Constraints

$$\wr : \mathbb{N}^2 \to \mathbb{N}^2 \to \mathbb{P}$$

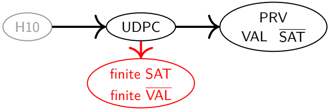
$$(a, b) \wr (c, d) := \underbrace{a + b + 1 = c}_{\text{Encodes addition}} \wedge \overbrace{\underbrace{d + d = b^2 + b}_{\text{Gaussian sum}}}^{\text{Encodes squaring}}$$

$$\frac{}{(a, 0) \wr (a + 1, 0)}$$

$$\frac{(d', b') \wr (d, d') \qquad (a, b') \wr (c', d') \qquad (c', 0) \wr (c, 0) \qquad (b', 0) \wr (b, 0)}{(a, b) \wr (c, d)}$$

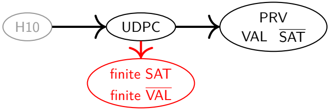
- $\wr$ as inductive relation axiomatizes itself
- $\wr$ can encode any equation on $\mathbb{N}$
- UDPC: Given set of constraints of shape $\wr$, is there a solution?
  - Undecidable by reduction from H10

finite VAL and the small fragment



Into the $(\forall, \to, \bot)$-fragment?

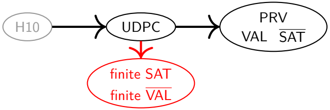
finite VAL and the small fragment



Into the $(\forall, \rightarrow, \bot)$-fragment?

- Finite models **behave classically**: $M \vDash \varphi$ decidable

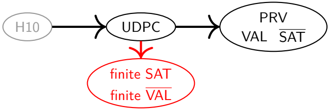
finite VAL and the small fragment



Into the $(\forall, \rightarrow, \bot)$-fragment?

- Finite models **behave classically**: $M \vDash \varphi$ decidable
- Negative translation works in general

finite VAL and the small fragment

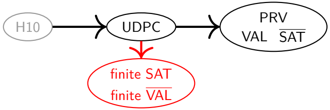


Into the $(\forall, \rightarrow, \bot)$-fragment?

- Finite models **behave classically**: $M \vDash \varphi$ decidable
- Negative translation works in general
- Finite SAT is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

What about finite VAL?

finite VAL and the small fragment



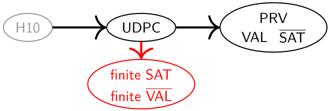
Into the $(\forall, \rightarrow, \bot)$-fragment?

- Finite models **behave classically**: $M \vDash \varphi$ decidable
- Negative translation works in general
- Finite SAT is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

What about finite VAL?

- Negate old reduction function
- Finite VAL is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

finite VAL and the small fragment



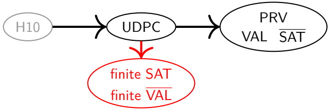
Into the $(\forall, \rightarrow, \bot)$-fragment?

- Finite models **behave classically**: $M \vDash \varphi$ decidable
- Negative translation works in general
- Finite SAT is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

What about finite VAL?

- Negate old reduction function
- Finite VAL is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Conjecture: finite VAL undecidable for dyadic signature over $(\forall, \rightarrow)$-fragment

finite VAL and the small fragment



Into the $(\forall, \rightarrow, \bot)$-fragment?

- Finite models **behave classically**: $M \vDash \varphi$ decidable
- Negative translation works in general
- Finite SAT is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment

What about finite VAL?

- Negate old reduction function
- Finite VAL is **undecidable** for dyadic signature over $(\forall, \rightarrow, \bot)$-fragment
- Conjecture: finite VAL undecidable for dyadic signature over $(\forall, \rightarrow)$-fragment
  - Friedman translation should be possible
  - Likely to require expanded standard model