Undecidability of Dyadic First-Order Logic in Coq

Johannes Hostert
Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany

Andrej Dudenhefner
Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany

Dominik Kirst
Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany

Abstract

We develop and mechanize compact proofs of the undecidability of various problems for dyadic first-order logic over a small logical fragment. In this fragment, formulas are restricted to only a single binary relation, and a minimal set of logical connectives. We show that validity, satisfiability, and provability, along with finite satisfiability and finite validity are undecidable, by directly reducing from a suitable binary variant of Diophantine constraints satisfiability. Our results improve upon existing work in two ways: First, the reductions are direct and significantly more compact than existing ones. Secondly, the undecidability of the small logic fragment of dyadic first-order logic was not mechanized before. We contribute our mechanization to the Coq Library of Undecidability Proofs, utilizing its synthetic approach to computability theory.

2012 ACM Subject Classification Theory of computation → Constructive mathematics; Theory of computation → Type theory; Theory of computation → Logic and verification

Keywords and phrases undecidability, synthetic computability, first-order logic, Coq

Digital Object Identifier 10.4230/LIPIcs.ITP.2022.19

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

Acknowledgements We thank Yannick Forster for valuable feedback on drafts of this paper.

1 Introduction

In 1928, Hilbert and Ackermann [18] mused whether there is a general decision procedure for the *Entscheidungsproblem*, that is, the problem of whether or not a formula of first-order logic is valid in all models. In the following years, several strategies were developed to approach this problem. So-called *reduction theory* [20, 3, 31] tried to reduce the general Entscheidungsproblem to simpler classes of first-order formulas. For formulas over a *monadic signature*, that is, formulas where all function and relation symbols are unary, Löwenheim [26] already established in 1915 that the Entscheidungsproblem is decidable.

Validity for first-order formulas was first shown undecidable for the general case of an unrestricted signature in 1936 by both Church [4] and Turing [34]. Shortly afterwards, in 1937, Kalmár [20] finalized a validity-preserving reduction chain allowing to convert a general first-order formula to one over a *dyadic signature*, that is, one with only a single binary relation symbol (and no function symbols). The resulting (un)decidability classification regarding the signature is as follows:

▶ **Theorem** ((Un-)decidability of validity)**.** Validity for first-order formulas is, depending on the arity of the function and relation symbols in the signature:

1. decidable, if all function and relation symbols are at most unary.
2. decidable, if all relation symbols are nullary.
3. undecidable, if there is an at least binary relation symbol.
4. undecidable, if there is an at least binary function, and a non-nullary relation symbol.

Proof summary.

1. By Löwenheim [26].
2. Quantifiers can be ignored, and thus formulas degenerate into trivial propositional logic.
3. By Kalmár [20], refining Church [4] and Turing [34].
4. By Gurevich [16]. ◀

Seminal works in reduction theory by Kalmár [21, 20] and Ackermann [1] in 1932-1939, which minimized the signature and the quantifier prefix, as well as techniques developed by Gödel in 1929 [14] and Gentzen in 1936 [13], which minimized the fragment of logical connectives, soon established several axes along which formula classes could be determined decidable or undecidable. These mentioned restrictions are all *syntactic*, i.e. they classify formulas by restricting their syntactic formation rules. Complementary to this are *semantic* restrictions, like the one emerging from finite model theory, where validity and satisfiability only involve finite models.

The undecidability of *finite satisfiability*, which is the variant of the Entscheidungsproblem restricted to finite models, was first shown in 1950 by Trakhtenbrot [33], who reduced from a semantic problem on μ-recursive functions. Although Trakhtenbrot's result holds for general first-order formulas, it was already folklore in 1950 that this result can be reduced to the undecidability of finite dyadic first-order logic (e.g. Kalmár [20] already claims that his reduction applies here, albeit without proof). In general, the above theorem applies to finite validity (and satisfiability) mutatis mutandis, as do the results for minimizing the fragment of logical connectives.

In recent years, most of these results have been mechanized in proof assistants, mainly Coq [32]. Notably, a classical mechanization of undecidability results would start by mechanizing a model of computation as a reference point for undecidability, which is cumbersome as it involves programming low-level entities of this model. Alternatively, one can use a synthetic approach to computability theory, which uses the notion of computation implicit in constructive foundations such as the type theory underlying Coq. This approach was developed by Forster et al. [9, 8], building on ideas by Richman [30] and Bauer [2].

Using this framework, Forster et al. [9] mechanize the undecidability of the Entscheidungsproblem in Coq. As they work in a constructive setting logic, they need to separately consider the undecidability of validity, satisfiability, and provability, as the various theorems establishing their many-one equivalence only hold classically. Following this approach, Kirst and Hermes [22] mechanize the undecidability of case (3) of the above theorem by transforming formulas of first-order ZF set theory, shown undecidable as well, into a signature with just a single binary relation $\in$. Kirst and Larchey-Wendling [23] mechanize the general case of Trakhtenbrot's result, as well as a signature compression chain and further decidability results to establish all cases of the above theorem for finite models. All of these results are collected in the Coq Library of Undecidability Proofs [11].

This aforementioned library also contains a mechanization of the undecidability of Diophantine constraints satisfiability, due to Larchey-Wendling and Forster [25]. This problem was first shown undecidable in 1970 by Matiyasevich [27], building upon work by Davis, Putnam, and Robinson [6] and provides the basis for reductions in this paper.

Contributions. Complementing existing results, we develop novel compact reductions establishing the undecidability of dyadic first-order logic. We develop a compactly mechanizable variant of Diophantine constraints satisfiability and directly reduce from it to show validity, satisfiability (for both Tarski and Kripke semantics), and intuitionistic and classical provability undecidable. We develop a similar reduction for finite satisfiability and validity.

Additionally, we strengthen our results over the existing mechanizations, namely by reducing the logical fragment to a minimal one. For some problems, this includes eliminating the falsity constant. In particular, we provide the first mechanization[1] that first-order validity and provability are undecidable for a dyadic signature and minimal logical fragment.

Outline. In Section 2, we recall the basics of synthetic undecidability and the mechanization of first-order logic we use. In Section 3, we introduce the binary variant UDPC of Diophantine constraints satisfiability we base our reductions on and show that it is undecidable. Section 4 contains the reductions from UDPC to validity, satisfiability, and provability, as well as the later minimization of the logical fragment. Section 5 does the same for finite satisfiability and validity, again including a mechanization for the minimal fragment. Finally, Section 6 summarizes our results and compares them to prior work.

2 Preliminaries

2.1 Synthetic Undecidability

We work in the Calculus of Inductive Constructions, a constructive type theory [5, 28]. Our type theory includes dependent functions $(\lambda(x : A).\,(e : B)) : \forall(x : A).\,B$, as well as inductive types, which include the empty type $\emptyset$, the unit type containing $\star : \mathbb{1}$, products $(a, b) : A \times B$, and sums $A + B$, as well as the natural numbers $\mathbb{N} := 0 \,|\, S\,n$ and booleans $\mathbb{B} := \mathtt{tt} \,|\, \mathtt{ff}$. Further, we have optionals $\mathcal{O}(X) := \emptyset \,|\, \lceil x \rceil$ and lists $\mathscr{L}(X) := [] \,|\, x :: xs$. A type is *listable* if there is a list including all elements of that type.

Type universes form a cumulative hierarchy $\mathbb{T}_0 : \mathbb{T}_1 : \cdots$, along with $\mathbb{P} : \mathbb{T}_1$, the computationally irrelevant type of propositions. By default, this implements an intuitionistic logic, where the law of excluded middle $\mathsf{LEM} := \forall P : \mathbb{P}.\,P \vee \neg P$ is not asserted. Since the hierarchy of types is cumulative, we omit indices wherever not necessary.

Since our type theory is constructive, functions defined in it are computable. In particular, this type system is implemented by the Coq proof assistant [32], which witnesses the computability and allows extraction to other programming languages.

Synthetic undecidability theory [8, 9] describes the approach for mechanizing undecidability proofs underlying the Coq library of Undecidability Proofs [11]. Due to the implicit computability of functions in this type theory, one can specify and verify computable functions without reference to a particular model of computation. For instance, a problem (a unary predicate) is decidable if there is a function computing its truth value. We refer the reader to the mentioned literature [8, 9] for a more detailed justification of the synthetic method.

▶ **Definition 1.** *Let $P : X \to \mathbb{P}$ be a problem on $X : \mathbb{T}$.*

- *The problem $\overline{P}$ such that $\overline{P}\,x := \neg P\,x$ is the* complement *of P.*
- *A function $f : X \to \mathbb{B}$ is a* decider for *P iff $\forall x : X.\,Px \leftrightarrow fx = \mathtt{tt}$.*
- *A function $f : \mathbb{N} \to \mathcal{O}(X)$ is an* enumerator for *P iff $\forall x : X.\,Px \leftrightarrow \exists n.\,fn = \lceil x \rceil$.*
- *P is decidable, written $\mathsf{dec}(P)$, iff there is a decider for P.*
- *P is enumerable, written $\mathsf{enum}(P)$, iff there is an enumerator for P.*

In particular, we are able to mechanize many-one reductions, verify them, and thereby establish that a problem is (synthetically) undecidable if we can many-one reduce to it from the halting problem of the λ-calculus, which is known to be undecidable [4]. We refer to

[1] Accessible from the supplementary web page and hyperlinked with every highlighted statement.

this halting problem as $HALT_\lambda$. Note that while $HALT_\lambda$ is also defined on open terms, it is many-one equivalent to a more canonical version of the halting problem on closed λ-expressions [8].

▶ **Definition 2.** *A problem $P : X \to \mathbb{P}$ on $X : \mathbb{T}$ is undecidable iff* $\mathsf{dec}(P) \to \mathsf{enum}(\overline{HALT_\lambda})$.

▶ **Definition 3.** *A problem $P : X \to \mathbb{P}$ on $X : \mathbb{T}$ many-one reduces to $Q : Y \to \mathbb{P}$, written $P \preccurlyeq Q$, iff there is a function $f : X \to Y$ such that $\forall x : X.\, Px \leftrightarrow Q(fx)$.*

When verifying a reduction, we typically call the $\to$-step preservation and the $\leftarrow$-step reflection.

▶ **Fact 4.** *If $\overline{HALT_\lambda}$ or $HALT_\lambda$ many-one reduces to P, then P is undecidable.*

2.2 First-Order Logic

First-order logic (FOL) is a logic where quantifiers may only range over individuals of the model, and not over functions or relations on these individuals.

We work within the mechanization of first-order logic already given in the Coq Library of Undecidability Proofs [11], which was synthesized by Kirst and Hermes [22] from previous work by Forster et al. [9, 10] and Kirst and Larchey-Wendling [23]. We recollect their definition here for ease of access:

First, fix a signature $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$ of function symbols $f : \mathcal{F}_\Sigma$ and relation symbols $P : \mathcal{P}_\Sigma$ with arities $|f|$ and $|P|$, which are then used to describe terms $t : \mathcal{T}$ and formulas $\varphi : \mathcal{F}$ as inductive types:

$$t ::= \mathsf{x}_n \mid f\,\vec{t} \quad (n : \mathbb{N}, \vec{t} : \mathcal{T}^{|f|}) \qquad \varphi ::= P\,\vec{t} \mid \bot \mid \varphi \to \psi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \forall\varphi \mid \exists\varphi \quad (\vec{t} : \mathcal{T}^{|P|})$$

While this definition and the mechanization uses de Bruijn indices to implement binding [7], we will use named binders on paper to improve readability.

Next, define the usual Tarski semantics providing an interpretation of formulas:

▶ **Definition 5.** *A* model $\mathcal{M}$ *consists of a domain type D as well as functions $f^{\mathcal{M}} : D^{|f|} \to D$ and $P^{\mathcal{M}} : D^{|P|} \to \mathbb{P}$ interpreting the symbols in the signature Σ. Given a variable assignment $\rho : \mathbb{N} \to D$ we define* term evaluation $\hat{\rho} : \mathcal{T} \to D$ *and* formula satisfaction $\rho \vDash \varphi$ *by*

$$\hat{\rho}\,\mathsf{x}_n := \rho\, n \qquad \hat{\rho}\,(f\,\vec{t}) := f^{\mathcal{M}}\,(\hat{\rho}\,\vec{t}) \qquad\qquad \rho \vDash P\,\vec{t} \;:=\; P^{\mathcal{M}}(\hat{\rho}\,\vec{t})$$

where the remaining cases of $\rho \vDash \varphi$ map each logical connective to its meta-level counterpart.

If a model $\mathcal{M}$ satisfies a formula φ for all variable assignments ρ, write $\mathcal{M} \vDash \varphi$.

We also follow Kirst and Hermes' [22] mechanization of provability, which is based on an inductive natural deduction system. We write $A \vdash \varphi$ for a list of formulas A and a formula φ if φ can be deduced from A in the intuitionistic deduction system, and $\vdash_c$ for the classical one. The full set of deduction rules can be found in [22].

In particular, these notions induce the following decision problems on formulas:

▶ **Problem 6** (Variants of the Entscheidungsproblem)**.**

- $\mathsf{VAL}\,\varphi := \forall\mathcal{M}.\, \forall\rho : \mathbb{N} \to D.\, \rho \vDash \varphi$
- $\mathsf{SAT}\,\varphi := \exists\mathcal{M}.\, \exists\rho : \mathbb{N} \to D.\, \rho \vDash \varphi$
- $\mathsf{PRV}\,\varphi := [] \vdash \varphi$
- $\mathsf{PRV}_c\,\varphi := [] \vdash_c \varphi$

When mechanizing finite model theory, we closely follow the mechanization developed by Kirst and Larchey-Wendling [23]. In particular, a finite model is not just listable, but also has decidable relations.

▶ **Definition 7.** A model $\mathcal{M}$ is *finite* iff D is listable and for all relation symbols p, $\mathsf{dec}(p^{\mathcal{M}})$ holds.

▶ **Problem 8** (Variants of the Entscheidungsproblem on finite models)**.**
- FVAL $\varphi := \forall \mathcal{M}.\, \mathcal{M}$ finite $\to \forall \rho : \mathbb{N} \to D.\, \rho \models \varphi$
- FSAT $\varphi := \exists \mathcal{M}.\, \mathcal{M}$ finite $\wedge\, \exists \rho : \mathbb{N} \to D.\, \rho \models \varphi$

3 Uniform Diophantine Pair Constraints

In order to achieve compact reductions to first-order logic, we need to pick a source problem which allows for easy formalization in first-order logic. Our particular problem is satisfiability for a variant of Diophantine constraints, that is, for a collection of equations on $\mathbb{N}$ all having a certain shape. The concrete shape is defined by the relation $\wr$.

▶ **Definition 9** ($\wr$)**.** *$\wr$ is the following relation on $\mathbb{N}^2 \times \mathbb{N}^2$:*

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

This relation has several suitable features. First of all, it is total and functional. It further allows encoding the successor operation, addition, and squaring, which suffices to express all of natural arithmetic. As Definition 12 shows, it can easily be characterized inductively. Intuitively, this relation encodes the Gaussian sum $d = \frac{b^2+b}{2} = \sum_{i=1}^{b} i$.

To define a constraints collection, we pick $\mathbb{N}$ as a concrete, countably infinite, discrete type of variables $\mathcal{V}$ and define:

▶ **Definition 10** (Uniform Diophantine Pair Constraints)**.** *A* uniform Diophantine pair constraint *is a tuple $((x, y), (z, w))$, with $x, y, z, w : \mathcal{V}$. An assignment $\rho : \mathcal{V} \to \mathbb{N}$* satisfies *such a constraint $((x, y), (z, w))$ iff $(\rho\, x, \rho\, y) \wr (\rho\, z, \rho\, w)$.*

▶ **Problem 11** (UDPC)**.** UDPC *is the following problem:*

$$\mathsf{UDPC}\,(h : \mathscr{L}\left(\mathcal{V}^2 \times \mathcal{V}^2\right)) := \exists \rho. \forall ((x, y), (z, w)) \in h.\, \rho \text{ satisfies } ((x, y), (z, w))$$

As mentioned, the problem also admits an inductive characterization:

▶ **Definition 12** (Inductive $\wr$)**.** *The relation $\wr$ can be equivalently characterized by:*

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

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

This characterization already hints at an axiomatic definition of $\wr$, as the axioms implied by the constructors are almost sufficient. The total axiomatization, consisting of the *Base*, *Step*, and *Tieback* axioms, is given here:

▶ **Lemma 13** (Basic properties of $\wr$)**.**
- Base*:* $(a, 0) \wr (c, 0)$ *iff* $c = a + 1$
- Step*:* $(a, b) \wr (c, d) \wedge b \neq 0$ *iff there are* b', c', d' *such that* $(a, b') \wr (c', d') \wedge (d', b') \wr (d, d') \wedge (b', 0) \wr (b, 0) \wedge (c', 0) \wr (c, 0)$
- Tieback*:* $(a, 0) \wr (c, d) \to d = 0$
- weak Step*:* $(a, b') \wr (c', d') \to (d', b') \wr (d, d') \to (b', 0) \wr (b, 0) \to (c', 0) \wr (c, 0) \to (a, b) \wr (c, d)$

Base and Step are stronger than the corresponding constructors of Definition 12, as they also capture elimination principles. The intuition behind the Step rule is that it encodes a Gaussian sum as we go from b' to $b'+1=b$: We must change $d'=\sum_{i=1}^{b'} i$ to $d=\sum_{i=1}^{b'+1} i$, so $d'+b'+1=d$ must hold, which is ensured by $(d',b')\wr(d,d')$.

▶ **Theorem 14** (Soundness and Completeness). *Any relation $\mathcal{R}$ on $\mathbb{N}^2\times\mathbb{N}^2$ satisfying Base, Step, and Tieback is equivalent to $\wr$. If $\mathcal{R}$ only satisfies Base and weak Step then completeness holds, while soundness does not necessarily do so anymore: only $(a,b)\wr(c,d)\to(a,b)\ \mathcal{R}\ (c,d)$ can be proven.*

▶ **Corollary 15**. *$\wr$ is the smallest relation satisfying the Base and weak Step axiom.*

▶ **Proposition 16** (Irreflexivity of $\wr$). $\forall(a,b):\mathbb{N}^2.\,\neg((a,b)\wr(a,b))$

To show UDPC undecidable, we reduce from a very similar problem called UDC, defined on uniform Diophantine constraints. An undecidability proof for this problem is already mechanized in the Coq Library of Undecidability Proofs.

▶ **Definition 17** (Uniform Diophantine Constraints). *A* uniform Diophantine constraint *is a triple (x,y,z), with $x,y,z:\mathcal{V}$. An assignment $\rho:\mathcal{V}\to\mathbb{N}$* satisfies *a uniform Diophantine constraint (x,y,z) iff*

$$1+\rho\,x+(\rho\,y)^2=\rho\,z$$

▶ **Problem 18** (UDC). UDC *is the following problem:*

$$\mathsf{UDC}\,(l:\mathscr{L}\left(\mathcal{V}^3\right)):=\exists\rho.\,\forall(x,y,z)\in l.\,\rho\ \textit{satisfies}\ (x,y,z)$$

▶ **Fact 19**. UDC *is undecidable.*

Fact 19 is mechanized by reducing from the general undecidability result for Diophantine constraints satisfiability by Larchey-Wendling and Forster [25]. Hence, our undecidability proof fundamentally relies on the general undecidability of Diophantine constraints satisfiability.

We can reduce from UDC to show UDPC undecidable. We only sketch the reduction.

▶ **Theorem 20**. $\mathsf{UDC}\preccurlyeq\mathsf{UDPC}$

Proof sketch. For each variable v appearing in the instance of UDC, have five new variables $v_i, 0\le i\le 4$. Then, for each constraint $1+x+y^2=z$, encode it using these new constraints:

$$(y_1,y_1)\wr(y_2,y_4),\quad (y_3,y_0)\wr(y_2,y_1),\quad (y_3,x_0)\wr(z_0,x_1)$$

The new variables v_i are thereby assigned the following values based on the old value of v:

i	0	1	2	3	4
v_i	v	$\frac{v^2+v}{2}$	v^2+v+1	v^2	$\frac{v_1^2+v_1}{2}$

◀

▶ **Theorem 21**. UDPC *is undecidable.*

Proof. By Theorem 20 and Fact 19. ◀

4 Undecidability of Validity

We now fix the concrete dyadic signature $\Sigma_{\wr}$, where the binary relation $\wr$ is the only symbol, and proceed to work within this signature unless explicitly mentioned.

4.1 Reducing from UDPC

For our reduction, we are given a constraints set $h : \mathscr{L}(\mathcal{V}^2 \times \mathcal{V}^2)$, and have to construct a formula $F^{\mathsf{VAL}}\, h$ such that $F^{\mathsf{VAL}}\, h$ is valid in all models if and only if h had a solution – formally, $\mathsf{UDPC}\, h \leftrightarrow \mathsf{VAL}\, (F^{\mathsf{VAL}}\, h)$. For this, we construct a first-order formalization of $\wr$, which later allows us to translate concrete constraints into FOL.

During the reflection step, we need to instantiate the proof that $F^{\mathsf{VAL}}\, h$ is valid with a specific model $\mathcal{M}_\wr$, which we develop now. Fix $D := \mathbb{N} + \mathbb{N}^2$ as the type of objects in $\mathcal{M}_\wr$. We can then define the interpretation.

▶ **Definition 22** (Interpretation of $\wr$ for $\mathcal{M}_\wr$).

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

Since we develop our axioms based on this model, we consider it the *standard model* of the following theory. We understand $\wr$ as a binary relation on pairs, and also add the numbers these pairs are made up from as individuals. Our interpretation of $\wr$ then also allows describing the pair's components. The definition for $x \wr y$ with $x, y : \mathbb{N}$ allows discriminating pairs and numbers, since $\wr$ (on pairs) is irreflexive (Proposition 16).

We can then begin axiomatizing $\wr$, starting with a few shorthands:

$$N\, k := k \wr k \qquad P\, p\, l\, r := \neg N\, p \land N\, l \land N\, r \land l \wr p \land p \wr r$$
$$R\, a\, b\, c\, d := \exists pq. P\, p\, a\, b \land P\, q\, c\, d \land p \wr q$$

In the standard model, $N\, k$ iff k is a number, and $P\, k\, l\, r$ iff k is the pair (l, r). $R\, a\, b\, c\, d$ is satisfied iff $(a, b) \wr (c, d)$.

We can now formalize our axioms in first-order logic. Due to Theorem 14, the Base and weak Step axioms are sufficient for the purpose of mechanizing this reduction.

Additionally, since our original relation is built on the linear order of natural numbers, we need to mechanize enough properties of natural numbers to re-establish this linear order in our relation. Thus, we first add axioms characterizing the natural numbers.

$$A_1^{\mathsf{VAL}} := \forall k. N\, k \to \exists k'. R\, k\, \dot{0}\, k'\, \dot{0} \qquad A_2^{\mathsf{VAL}} := N\, \dot{0}$$

Axioms A_1^{VAL} and A_2^{VAL} postulate that $\dot{0}$ is a natural number, and that there are successors. The encoding of $S\, x = y$ is $(x, \dot{0}) \wr (y, \dot{0})$, implicitly encoding the Base axiom.

$$A_3^{\mathsf{VAL}} := \forall abcdb'c'd'. R\, a\, b'\, c'\, d' \land R\, d'\, b'\, d\, d' \land R\, b'\, \dot{0}\, b\, \dot{0} \land R\, c'\, \dot{0}\, c\, \dot{0} \to R\, a\, b\, c\, d$$

Axiom A_3^{VAL} formalizes the weak Step axiom.

Throughout these axioms, we have used $\dot{0}$ as if it was a nullary function symbol (i.e. a constant). In order to keep our signature minimal, we add this as an outermostly quantified variable to our formula. We call such a construction a *mock constant*.

We can now construct our reduction function:

$$F^{\mathsf{VAL}}, \mathit{code} : \mathscr{L}\left(\mathcal{V}^2 \times \mathcal{V}^2\right) \to \mathcal{F}$$
$$F^{\mathsf{VAL}}\, h := \forall \dot{0}. A_1^{\mathsf{VAL}} \to A_2^{\mathsf{VAL}} \to A_3^{\mathsf{VAL}} \to \exists_{v \in \mathcal{V}(h)}\, \mathit{code}\, h$$
$$\mathit{code}\, [] := \top$$
$$\mathit{code}\, (((a, b), (c, d)) :: h) := R\, a\, b\, c\, d \land \mathit{code}\, h$$

Besides ensuring the axioms hold, the reduction function encodes the *satisfaction condition* using one ∃-quantifier per variable in the constraint set h, thereby requiring that the model has a solution satisfying the first-order-encoded constraints.

▶ **Lemma 23** (Reflection). $\mathsf{VAL}\,(F^{\mathsf{VAL}}(h)) \to \mathsf{UDPC}\,h$

Proof. We have that $F^{\mathsf{VAL}}(h)$ is valid in all models, so it in particular is satisfied by $\mathcal{M}_\wr$, which satisfies A^{VAL}_{1-3}. The satisfaction condition ensures the model "knows" a solution, which we can extract since the interpretation of $\wr$ in $\mathcal{M}_\wr$ is faithful. ◀

For preservation, we are given a solution ρ satisfying h, and have to reason in an abstract model. By construction of F^{VAL}, we can assume the axioms A^{VAL}_{1-3} and must now instantiate a solution for the satisfaction condition. To do so, we first need to find the elements in our model corresponding to natural numbers. For this, we define a data structure called "chain":

▶ **Definition 24** (Chain). *A* chain *up to* $m : \mathbb{N}$ *is a function* $f : \mathbb{N} \to D$ *such that all of*
1. $f\,0 = \dot{0}$
2. *for all* $n < m$*, we have* $R\,(f\,n)\,\dot{0}\,(f\,(n+1))\,\dot{0}$

If $f\,n = d$*, then* d represents n.

Such a chain is just a partial function that maps elements of $\mathbb{N}$ to their representatives in the model.

Starting in Definition 24, we abuse notation by using first-order formulas and terms to denote properties in and elements of the model.

▶ **Proposition 25**. *Given a chain* f *up to* m *and an* $n \le m$*, we have* $N\,(f\,n)$.

Proof. For 0, use A^{VAL}_2. Otherwise, by definition of R. ◀

In order to give a proof of $M \vDash \exists_{v \in \mathcal{V}(h)}\, code\, h$, we need to first construct a chain:

▶ **Proposition 26** (Chain construction). *Given* m*, there exists a chain* f *up to* m.

Proof. Induction on m:
- $m = 0$: The chain is given by $f\,n := \dot{0}$. This satisfies both properties, using Proposition 25.
- $m = m' + 1$, where f' is a chain up to m' by induction. We apply A^{VAL}_1 to $f'\,m'$ and get d such that $R\,(f'\,m')\,0\,d\,0$. Our chain f up to m is chosen as $f'[m \mapsto d]$. Chain property 1 is shown by induction, and property 2 also is using Proposition 25, except for $n = m'$, where it is satisfied because A^{VAL}_1 gave us d already fulfilling this required property. ◀

We now use Proposition 26 to build a chain f up to $\max_{v \in \mathcal{V}(h)} \rho\, v$. Then, we can proceed to prove the satisfaction condition by instantiating as follows: For a variable v, chose $f\,(\rho\, v)$. The remaining goals are now of shape $R\,(f\,(\rho x))\,(f\,(\rho y))\,(f\,(\rho z))\,(f\,(\rho w))$, for all $((x, y), (z, w)) \in h$. Finally, this can be shown using another lemma:

▶ **Proposition 27** (Chain steps). *If* $(a, b) \wr (c, d)$*,* $a, b, c, d < m$ *and* f *is a chain up to* m*, then* $M \vDash R\,(f\,a)\,(f\,b)\,(f\,c)\,(f\,d)$.

Proof. Induction on the inductive characterization of $(a, b) \wr (c, d)$:
- $a + 1 = c, b = d = 0$: Since $a < m$ and f is a chain, we are done by the chain properties.
- We have $R\,(f\,a)\,(f\,b')\,(f\,c')\,(f\,d')$, $R\,(f\,d')\,(f\,b')\,(f\,d)\,(f\,d')$, $R\,(f\,b')\,(f\,\dot{0})\,(f\,b)\,(f\,\dot{0})$, and $R\,(f\,c')\,(f\,\dot{0})\,(f\,c)\,(f\,\dot{0})$ by induction as $a, b, c, d, b', c', d' < n$ holds. We can apply A^{VAL}_3 and are done. ◀

With this lemma, we can conclude the complete proof of $M \vDash F^{\mathsf{VAL}}\,h$ for an arbitrary M.

▶ **Lemma 28** (Preservation). $\mathsf{UDPC}\, h \to \mathsf{VAL}\, (F^{\mathsf{VAL}}(h))$

Proof. We have ρ, a solution to h, and the fact that our model fulfills the axioms A^{VAL}_{1-3}. Thus, we can build a chain f up to $\max_{v \in \mathcal{V}(h)} \rho\, v$ by Proposition 26. This chain allows us to instantiate the satisfaction condition, concretely using $f(\rho\, v)$ for given $v : \mathcal{V}$. We conclude by Proposition 27 for the remaining goals created by *code*. ◀

▶ **Theorem 29**. $\mathsf{UDPC} \preccurlyeq \mathsf{VAL}$ *restricted to dyadic formulas.*

Proof. F^{VAL} is a reduction function by Lemma 23 and Lemma 28. ◀

▶ **Theorem 30**. $\overline{\mathsf{UDPC}} \preccurlyeq \mathsf{SAT}$ *restricted to dyadic formulas.*

Proof. Using $F'\, \varphi := \neg(F^{\mathsf{VAL}}\, \varphi)$ as reduction function. ◀

4.2 Minimizing the Logical Fragment

Next, we adapt the just outlined reduction to not only witness the undecidability of validity over the minimal signature, but to also establish the undecidability over a[2] minimal set of logical operators, namely the forall-implicative fragment.

▶ **Definition 31** (Forall-implicative fragment). *A formula using only $\bot$, the logical connective $\to$, and the $\forall$-quantifier is within the* $(\forall, \to, \bot)$-fragment.

▶ **Definition 32** (Negation). *A formula in the $(\forall, \to, \bot)$-fragment not containing $\bot$ is within the* $(\forall, \to)$-fragment. *In general, a formula without $\bot$ is in a fragment* without negation.

We do so employing a mostly standard translation process, which combines ideas of both Gödel-Gentzen double negation translation [14, 13] and Friedman's A-translation [12], as outlined by Forster et al. [9]. Put simply, we apply a double-negation translation, and replace all uses of $\bot$ by another formula. The resulting formula then is within the $(\forall, \to)$-fragment.

This transformation has not yet been mechanized to hold in general, so we manually apply it to our concrete formula, and construct a new reduction with this reduced formula.

Further, the transformation usually introduces a new relation symbol for $\bot$, which is not an option here, as we want to keep the signature minimal. Instead, we slightly change our standard model, exploiting as-of-yet unused space. To adapt our reduction, we define new syntactic sugar, subsuming the previous definitions.

$$\bot_w := c_1 \not\approx c_2 \qquad \bot_s := \forall ab.a \not\approx b \qquad \neg_w \varphi := \varphi \to \bot_w$$
$$N\, k := k \not\approx k \qquad P\, p\, l\, r\, \psi := (N\, p \to \bot_s) \to N\, l \to N\, r \to l \not\approx p \to p \not\approx r \to \psi$$
$$R\, p\, q\, a\, b\, c\, d\, \psi := P\, p\, a\, b\, (P\, q\, c\, d\, (p \not\approx q \to \psi))$$

There are two versions of $\bot$: $\bot_s$ and $\bot_w$, where $\bot_w$ is the canonical replacement of falsity. $\bot_s$ is actually false in our standard model, whereas $\bot_w$ will not be, and this allows us to simplify verifying the reduction:

A canonical Friedman A-translation would just replace all occurrences of $\bot$ by $\bot_w$ and insert sufficient double negations. Since working in doubly negated contexts is often cumbersome, our reduction aims to avoid this as much as possible. Using $\bot_s$, which is actually equivalent to falsity in the standard model, helps, since it makes the elimination

[2] There is no canonical unique minimal set of logical operators. We use $(\forall, \to)$, following Forster et al. [9], while Gentzen [13] used $(\forall, \wedge)$. In either case, a quantifier and a binary connective is necessary.

lemmas not be doubly negated, thereby eliding a significant amount of double negations. For instance, we can show that in $\mathcal{M}$, $(N\,p \to \bot_s)$ implies that p is a pair, and not just that $\neg_w\neg_w(p \text{ is a pair})$.

Along with this, we also translate our axioms:

$$A_2^\bot := N\,0 \qquad A_3^\bot := \forall abcdb'c'd'p_1 \ldots p_8.$$
$$A_1^\bot := \forall k.\, N\,k \to \neg_w \forall pqk'.\, R\,p\,q\,k\,\dot{0}\,k'\,\dot{0} \bot_w \qquad R\,p_1\,p_2\,a\,b'\,c'\,d'\ (R\,p_3\,p_4\,d'\,b'\,d\,d'$$
$$(R\,p_5\,p_6\,b'\,0\,b\,0\ (R\,p_7\,p_8\ c'\,0\,c\,0$$
$$(\neg_w \forall p_9 p_{10}.R\,p_9\,p_{10}\,a\,b\,c\,d\,\bot_w))))$$

Axiom 3 becomes larger as we have changed the definition of R. Previously, this hid away two pairs each, which are now explicitly universally quantified. This again serves to make our reduction easier to verify. The full reduction function is as follows:

$$F^\bot, \mathit{code} : \mathscr{L}\left(\mathcal{V}^2 \times \mathcal{V}^2\right) \to \mathcal{F}$$
$$F^\bot\,h := \forall \dot{0}\,c_1\,c_2.\, A_1^\bot \to A_2^\bot \to A_3^\bot \to \neg_w \bigvee_{v \in \mathcal{V}(h)} \mathit{code}\,h$$
$$\mathit{code}\,[] := \bot_w$$
$$\mathit{code}\,(((a,b),(c,d)) :: h) := \neg_w(\forall p_1 p_2.R\,p_1\,p_2\,a\,b\,c\,d\,\bot_w) \to \mathit{code}\,h$$

We need to slightly adjust our interpretation to make the Friedman translation work.

▶ **Definition 33** (Interpretation of ⟅ for Friedman-translation).

l \ r	$y \in \mathbb{N}$	$(c,d) \in \mathbb{N}^2$
$x \in \mathbb{N}$	$x = y \vee \underline{(x = 0 \wedge y = 1 \wedge \mathsf{UDPC}\,h)}$	$x = c$
$(a,b) \in \mathbb{N}^2$	$y = b$	$(a,b) \wr (c,d)$

Here, the aforementioned encoding of reified falsity is underlined. The encoding stipulates that $0 \wr 1 \Leftrightarrow \mathsf{UDPC}\,h$. This becomes relevant during reduction reflection, as it allows us to "break out" of the double-negated context by instantiating the mock constants c_1, c_2 with $c_1 := 0, c_2 := 1$. Intuitively[3], when replacing $\bot$ with A when Friedman-translating φ, the resulting formula is equivalent to $\varphi \vee A$. Thus, by choosing $0 \wr 1$ as A, we are able to extract a solution as we did before, and if we are not, we still have our result.

▶ **Proposition 34** (Proposition 26 for the minimal fragment). *Let $m : \mathbb{N}$. To show $\bot_w$, it suffices to show $\forall f.(f$ is a chain up to $m) \to \bot_w$.*

▶ **Proposition 35** (Proposition 27 for the minimal fragment).
If $(a,b) \wr (c,d)$, $a,b,c,d < m$ and f is a chain up to m, then showing $M \vDash \bot_w$ requires showing $M \vDash \forall pq.R\,p\,q\,(f\,a)\,(f\,b)\,(f\,c)\,(f\,d)\,\bot_w$.

Proposition 34 is the double-negated version of Proposition 26. Instead of posing the existence of a chain f, we allow adding hypotheses when aiming to prove $\bot_w$, which is equivalent under double negation. Proposition 35 is translated similarly.

Proving these propositions is very technical, as one has to work in a doubly-negated context most of the time. As mentioned, the axioms and the syntactic sugar we defined before aim to minimize double negations, which explains most of the unorthodox constructions (e.g. having both $\bot_w$ and $\bot_s$). With these two lemmas, our reduction is complete.

[3] This intuition only holds in classical logic, but is still useful for conceptualizing the translation

▶ **Theorem 36.** UDPC $\preccurlyeq$ VAL *restricted to dyadic formulas over the* $(\forall, \to)$*-fragment.*

Proof. Similar to Theorem 29, except for the changes outlined in this chapter. ◀

▶ **Theorem 37.** $\overline{\text{UDPC}} \preccurlyeq$ SAT *restricted to dyadic formulas over* $(\forall, \to, \bot)$*-fragment.*

Proof. Using $F'\,\varphi := \neg(F\,\varphi)$ as reduction function. ◀

SAT is trivially decidable over the $(\forall, \to)$-fragment, since one can construct a model where everything is true. It thus becomes undecidable as soon as there is a single use of $\bot$.

4.3 Undecidability of Provability

We now turn towards the undecidability of the provability predicate PRV.

In a classical meta-theory, this would directly follow from completeness [15]. In our intuitionistic meta-theory, however, this does not trivially hold [10, 24], and we instead have to manually mechanize our reduction for the deduction system. The reduction function stays the same, but instead of reasoning in an abstract model, we prove that the formula is deducible in our deduction system. This mainly impacts the preservation step, as we can use soundness for the reflection step:

▶ **Lemma 38** (Reflection). $\text{PRV}\,(F^{\bot}(h)) \to \text{UDPC}\,h$.

Proof. Immediate using Theorem 36 and soundness of the deduction system. ◀

For preservation, the proof structure follows the previous proofs, while the low-level goals change to accommodate the object-level deduction. For brevity, we only show the new definition of a chain, as it demonstrates changes necessary to construct a syntactic proof in the first-order deduction system.

▶ **Definition 39** (Chain for provability). *A* proto-chain up to n *is a list on* $\mathcal{V}^3$ *of length* n*, containing triples* (m, l, r)*. Every proto-chain* c *has a* head number *head* c*, which is* $\dot{0}$ *for* $[]$ *and* m *for* $[(m, l, r), \ldots]$*. The* chain hypotheses *hyp* $n\,c : \mathscr{L}(\mathcal{F})$ *for a proto-chain* c *up to* n *are defined inductively:*

$$\begin{aligned} hyp\,0\,[] &:= [N\,\dot{0}] \\ hyp\,(n'+1)\,((m,l,r) :: cr) &:= N\,m :: (N\,l \to \bot_s) :: (N\,r \to \bot_s) \\ &\quad :: l \between (head\,cr) :: \dot{0} \between l :: r \between m :: \dot{0} \between r :: l \between r :: hyp\,n'\,cr \end{aligned}$$

The definition now strongly separates data (the objects in our chain) and hypotheses this data must satisfy. Compared to this, Definition 24 kept the data implicit: for a number $n : \mathbb{N}$, we had $m = f\,n$, and the proof that f is a chain at n. That property (being a chain) is defined by an existential quantifier, and so a proof of it contained two pairs l, r as witnesses. This was possible since Tarski models directly embed into the meta-logic, so first-order existentials are represented as meta-level existentials, which just are dependent pairs (truncated into $\mathbb{P}$). For provability, these objects are now explicitly stored as the chain entry (m, l, r), as working within the deduction system prevents us from building a data structure (like dependent pairs) containing both these objects and the proofs about them.

Proposition 34 now looks like this, for instance:

▶ **Proposition 40** (Proposition 34 for PRV).
Let $m : \mathbb{N}$ *and* $A : \mathscr{L}(\mathcal{F})$ *be a list of hypotheses including* $A^{\text{VAL}^{\bot}}_{1-3}$*. To construct a proof* $A \vdash \bot_w$*, it suffices to construct a proof* $(hyp\,m\,c) + A \vdash \bot_w$ *for all proto-chains* c *up to* m*.*

This theorem builds a proof for $\bot_w$ from another proof for $\bot_w$ with more assumptions. Specifically, it allows us to assume that there is some proto-chain, and to also assume that it actually fulfills the chain hypotheses. As mentioned, data (the variables defining the chain) and the hypotheses about this data are maintained separately, since the hypotheses are part of the object-level deduction, while the data is not. Proposition 35 and the other lemmas from before undergo similar changes, which we omit for brevity.

▶ **Lemma 41** (Preservation). $\mathsf{UDPC}\, h \to \mathsf{PRV}\,(F^{\bot}(h))$.

Proof. Using the reformulated variants of Proposition 34 and Proposition 35, as outlined above, following the approach of Lemma 28. ◀

With this, we have shown that validity, satisfiability, and provability are undecidable for the minimal signature and minimal logical fragment.

▶ **Theorem 42.** $\mathsf{UDPC} \preccurlyeq \mathsf{PRV}$ *restricted to dyadic formulas over the* $(\forall, \to)$*-fragment. Assuming* LEM*, this holds for* PRV_c*.*

Proof. $F^{\bot}$ is a reduction function by Lemmas 38 and 41. For PRV_c, we need LEM to establish soundness in Lemma 38. ◀

Related Decision Problems. From the undecidability of provability, we additionally get the undecidability of Kripke validity (KVAL) and Kripke satisfiability (KSAT). Since we do not work in Kripke models, we refrain from formally introducing them here, and refer to Forster et al. [9] and Herbelin and Lee [17].

▶ **Theorem 43.** $\mathsf{UDPC} \preccurlyeq \mathsf{KVAL}$ *restricted to dyadic formulas over the* $(\forall, \to)$*-fragment.*

▶ **Theorem 44.** $\overline{\mathsf{UDPC}} \preccurlyeq \mathsf{KSAT}$ *restricted to dyadic formulas over the* $(\forall, \to, \bot)$*-fragment.*

5 Undecidability of Finite Satisfiability

For finite satisfiability (FSAT), we need to construct a new reduction function. The axioms used so far allow us to construct infinitely many numbers, while the reduction requires us to transport arbitrary large solution assignments. Both of these become impossible when working with finite models, potentially requiring fragile upper bounds on model size. Additionally, only $\overline{\mathsf{UDPC}}$ many-one reduces to FVAL, as both are co-recursively enumerable. Thus, a more straightforward approach is a reduction from UDPC towards FSAT. Such reductions work "inversely" in that the axioms typically resemble elimination schemes, instead of constructors.[4] Our axioms then merely deconstruct larger values into smaller ones, thus side-stepping the issue of making the axioms work without conflicting with the finiteness of the model.

The following reduction often needs to decide whether an arbitrary first-order formula is satisfied in some finite model. Since we required decidable relation interpretations, we can construct a general decider, as noted in [23]:

[4] Explicitly noted and discussed in [23]

▶ **Proposition 45** (Decidability of finite satisfaction). *Given a fixed finite model $\mathcal{M}$ and a fixed environment $\rho : \mathcal{V} \to \mathcal{M}$, the predicate $\mathcal{M} \vDash_\rho \varphi$ on $\varphi : \mathcal{F}$ is decidable.*

Proof. All relation symbol interpretations of $\mathcal{M}$ are decidable. Quantified formulas are decidable since finite quantification is decidable and $\mathcal{M}$ is finite. ◀

For the axioms, we need first-order indistinguishably $\equiv$ in order to guard our elimination axioms against constructing a predecessor of $\dot{0}$, since such a predecessor does not exist in our standard model. We can encode it as $a \equiv b := \forall k.a \between k \leftrightarrow b \between k \land k \between a \leftrightarrow k \between b$. The axioms of Lemma 13 are formalized as follows:

$$
\begin{aligned}
A_1^{\mathsf{FSAT}} &:= \forall k'.\, N\, k' \to k' \not\equiv \dot{0} \to \exists k.\, (k, \dot{0}) \between (k', \dot{0}) \\
A_2^{\mathsf{FSAT}} &:= \forall abcd.\, (a,b) \between (c,d) \to b \not\equiv \dot{0} \to \\
&\qquad \exists b'c'd'.\, (b', \dot{0}) \between (b, \dot{0}) \land (c', \dot{0}) \between (c, \dot{0}) \land (a, b') \between (c', d') \land (d', b') \between (d, d') \land d' < d \\
A_3^{\mathsf{FSAT}} &:= \forall aa'd.\, (a, \dot{0}) \between (a', d) \to d \equiv \dot{0}
\end{aligned}
$$

These axioms are based on the Base, strong Step, and Tieback laws respectively. We only need the "backwards" direction since we will only deconstruct a given relation.

These axioms alone are not sufficient to extract constraints solutions from a model. While these allow us to unpack a constraint into smaller constraints, the model might be cyclic, such that smaller constraints eventually unpack into themselves. To prevent this, we need to make the predecessor relation $(k, 0) \wr (k', 0)$ well-founded. While well-foundedness is not first-order expressible in general, it is possible to ensure that a relation on a finite model is well-founded:

▶ **Fact 46** (Well-founded relations for finite types). *Let D be a listable type and $\prec$ be a transitive, irreflexive relation on D. Then $\prec$ is well-founded.*

To make our predecessor relation well-founded, we need to construct its transitive closure. This is again not first-order expressible without adding a new relation symbol, which we can not do. Our solution is to exploit encoding space in the standard model in order to fit a new binary relation on numbers into the standard model. We define the following syntactic sugar, and add more axioms:

$$
\begin{aligned}
A_4^{\mathsf{FSAT}} &:= \forall lr.\, (l, \dot{0}) \between (r, \dot{0}) \to l < r \land \forall k.\, k < r \to k \le l & \qquad a < b &:= a \le b \land a \not\equiv b \\
A_5^{\mathsf{FSAT}} &:= \forall abc.\, a < b \to b < c \to a < c & \qquad a \le b &:= N\, a \land N\, b \land a \between b
\end{aligned}
$$

So $<$ becomes our transitive, irreflexive relation encompassing the predecessor relation. The actual reduction function can now be given. It also features an upper bound to ensure that we can build a sufficiently large anti-chain.

$$
\begin{aligned}
&F^{\mathsf{FSAT}}, \mathit{code} : \mathscr{L}(\mathcal{V}^2 \times \mathcal{V}^2) \to \mathcal{F} \\
&F^{\mathsf{FSAT}}\, h := \exists \dot{0} m.\, A_1^{\mathsf{FSAT}} \land A_2^{\mathsf{FSAT}} \land A_3^{\mathsf{FSAT}} \land A_4^{\mathsf{FSAT}} \land A_5^{\mathsf{FSAT}} \land \bigsqcap_{v \in \mathcal{V}(h)} \mathit{code}\, h \\
&\quad \mathit{code}\, [] := \top \\
&\quad \mathit{code}\, (((a,b),(c,d)) :: hs) := (a,b) \between (c,d) \land \mathit{code}\, hs \land a, b, c, d \le m
\end{aligned}
$$

Another mock constant is needed: m is an upper bound on the model size. Since we are reducing from satisfiability, mock constants must be existentially quantified. Reducing from FSAT also "flips" the reduction and preservation steps. We now start with preservation, which now features our standard model.

▶ **Lemma 47** (Preservation). $\mathsf{UDPC}\, h \to \mathsf{FSAT}\,(F^{\mathsf{FSAT}}(h))$

Proof. Given a solution ρ to a constraints set h, let $m := 1 + \max_{v \in \mathcal{V}(h)} \rho\, v$. We denote by $\mathbb{N}_{\leq m}$ the type of natural numbers up to (and including) m. We set our finite domain $D := \mathbb{N}_{\leq m} + \mathbb{N}^2_{\leq m}$, so that each element of the domain is either a natural number not larger than m, or the pair of two such numbers. To finalize our model $\mathcal{M}_{\leq m}$, we define the interpretation of ≀.

▶ **Definition 48** (Interpretation of ≀ for FSAT).

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

The model interpretation remains unchanged from the previous chapters, except for the top-left cell, which changes to $x \leq y$ from $x = y$. This is where we exploit the additional encoding space given to us by the fact last chapter (almost) only used the diagonal, where $x = y$. The interpretation of ≀ is also decidable, as is required for finite models.

Showing that the satisfaction condition holds also is straightforward: Choose $\rho\, v$ for each v, using that ≀ is faithfully interpreted. Each $\rho\, v$ is in $\mathbb{N}_{\leq m}$, since it is smaller than m by construction. Thus we have shown that $F^{\mathsf{FSAT}}(h)$ is finitely satisfied. ◀

For reflection, we need to extract a constraints collection solution from an arbitrary finite model $\mathcal{M} = (D, I)$ satisfying the axioms and the satisfaction condition. For this, we will construct an inverse notion of a chain, which now transports individuals of a model to natural numbers.

▶ **Definition 49** (Anti-chain). *A function $f : D \to \mathcal{O}(\mathbb{N})$ is called a* anti-chain up to $m : D$ representing $n : \mathbb{N}$ *iff all of*

1. $\forall d : D.\, d \leq m \Leftrightarrow f\, m \neq \emptyset$
2. $\forall k : \mathbb{N}.\, (\exists d : D. d \leq m \wedge f\, d = \lceil n \rceil) \Rightarrow k \leq n$
3. $f\, m = \lceil n \rceil \wedge f\, \left(\dot{0}\right) = \lceil 0 \rceil$
4. $\forall (d_l\, d_r : D)(k_l\, k_r : \mathbb{N}).\, f\, d_l = \lceil k_l \rceil \wedge f\, d_r = \lceil k_r \rceil \Rightarrow \left(S\, k_l = k_r \Leftrightarrow (d_l, \dot{0}) \wr (d_r, \dot{0})\right)$
5. $\forall d d'\, f.\, d \neq \emptyset \Rightarrow (f\, d = f\, d' \Leftrightarrow d \equiv d')$

We call m and n the upper bound of f*, and say that some m'* represents *some n' iff $f\, m' = \lceil n' \rceil$.*

▶ **Proposition 50** (Anti-chain construction). *Given $m : D$ such that $N\, m$, there are n, f such that f is an anti-chain up to m representing n.*

Proof. Well-founded induction using $<$ on m, due to Fact 46. We decide whether $m \equiv \dot{0}$ by Proposition 45:

- $m \equiv \dot{0}$. We choose as our anti-chain the function defined by $f\, \dot{0} = \lceil 0 \rceil$ and $f\, k = \emptyset$ otherwise. The chain properties are easily shown using decidability of $(\equiv)$.
- $m \not\equiv \dot{0}$. We know m has a predecessor m' by A_1^{FSAT}. Applying the induction hypothesis to m' yields f', an anti-chain representing m' up to n'. We choose $f'[k \mapsto \lceil n' + 1 \rceil]$ (i.e. pointwise updating f' at k) as our anti-chain. It is up to m representing $n' + 1$ by case distinctions, using the induction hypothesis and decidability of $(\equiv)$. ◀

Once we are able to construct an anti-chain, we are able to use it to extract solutions to the constraints encoded in h. For this, we show that $\wr$ on D transports to $\wr$ on $\mathbb{N}$ for the represented numbers.

▶ **Proposition 51** (Solution recovery). *If f is an anti-chain up to m representing n, and if $(a,b)\wr(c,d)$ where a,b,c,d are all $\leq m$, then we find a_r,b_r,c_r,d_r such that $f\,a=\lceil a_r\rceil$, $f\,b=\lceil b_r\rceil$, $f\,c=\lceil c_r\rceil$, $f\,d=\lceil d_r\rceil$, and $(a_r,b_r)\wr(c_r,d_r)$.*

Proof. Well-founded induction using $<$ on b with a,c,d quantified, using Fact 46.
We have $f\,a,\dots f\,d\neq\emptyset$ by anti-chain property 1. Again decide whether $b\equiv\dot{0}$:

- $b\equiv 0$. In this case, by A_3^{FSAT}, we find $d\equiv 0$. So we are given $(a,0)\wr(c,0)$, which fits anti-chain property 4 of f. Since $(x,0)\wr(x+1,0)$, we can conclude using properties 3 and 5.
- $b\not\equiv 0$. We can apply A_2^{FSAT}. We further apply the induction hypothesis to $(d',b')\wr(d,d')$ and $(a,b')\wr(c',d')$, which is possible especially since $d<d'$, which we needed to explicitly add to A_2^{FSAT}. The remainder is straightforward by the defining properties of $\wr$ and anti-chain property 4, similar to case 1. ◀

With this, our reduction is complete.

▶ **Lemma 52** (Reflection). $\mathsf{FSAT}(F^{\mathsf{FSAT}}(h))\to\mathsf{UDPC}\,h$

Proof. We need to first build an anti-chain f up to m using Proposition 50, where m is existentially quantified in F. After this, we are able to extract the numbers making up the solution to h by looking at the elements encoded in the satisfaction condition, and looking up the represented numbers in f. Afterwards, it remains to show that these numbers actually are solutions to h, which follows from Proposition 51 and some minor auxiliary lemmas. ◀

▶ **Theorem 53.** $\mathsf{UDPC}\preccurlyeq\mathsf{FSAT}$ *restricted to dyadic formulas.*

To reduce towards finite validity, simply negate the reduction function used for FSAT.

▶ **Theorem 54.** $\overline{\mathsf{UDPC}}\preccurlyeq\mathsf{FVAL}$ *restricted to dyadic formulas.*

5.1 Minimizing the Logical Fragment for FSAT

We now reduce the logical fragment for FSAT. Using Proposition 45, we can construct and verify a translation converting our formulas into the $(\forall,\to,\bot)$-fragment:

▶ **Definition 55** (Double negation translation). $(\cdot)^N$ *translates formulas $\varphi:\mathcal{F}$ to their double-negated counterpart in the $(\forall,\to,\bot)$-fragment. For instance, $(\varphi\vee\psi)^N=\neg\varphi^N\to\neg\neg\psi^N$ and $(\exists x.\,\varphi)^N=\neg(\forall x.\,\neg\varphi)$. The other cases are defined similarly.*

▶ **Proposition 56.** *Given a finite model $\mathcal{M}$, an environment $\rho:\mathcal{V}\to\mathcal{M}$, and a formula φ, $(M\vDash_\rho\varphi)\Leftrightarrow(M\vDash_\rho(\varphi)^N)$.*

Proof. By induction on φ, using Proposition 45 for $\wedge,\vee,\exists$. ◀

This now strengthens our previous undecidability result to a small logical fragment.

▶ **Theorem 57.** FSAT *reduces to* FSAT *over the $(\forall,\to,\bot)$-fragment, and* FVAL *similarly.*

Proof. The function $(\cdot)^N$ fulfills the reduction properties by Proposition 56. ◀

6 Conclusion

6.1 Summary of Results

We have shown that many common decision problems of FOL are undecidable in their minimal case by constructing many-one reductions from UDPC or $\overline{\text{UDPC}}$.

▶ **Theorem 58** (Undecidability of FOL problems). *The following problems are undecidable for first-order formulas with a single binary relation:*

	Problem	*Fragment*	*By reduction*
FOL	VAL	$(\forall, \to)$	*Theorem 36*
	SAT	$(\forall, \to, \bot)$	*Theorem 37*
	PRV	$(\forall, \to)$	*Theorem 42*
	PRV_c	$(\forall, \to)$	*Theorem 42 assuming* LEM
KFOL	KVAL	$(\forall, \to)$	*Theorem 43*
	KSAT	$(\forall, \to, \bot)$	*Theorem 44*
FFOL	FSAT	$(\forall, \to, \bot)$	*Theorems 53 and 57*
	FVAL	$(\forall, \to, \bot)$	*Theorems 54 and 57*

These results are minimal, except for finite validity (FVAL), where eliminating $\bot$ might be possible. However, we have not mechanized this, because our current reduction for FVAL occupies a local optimum where there is no space remaining in the interpretation of the standard model. Such space is however necessary for a Friedman-like falsity elimination, as done in Section 4.2. KFOL denotes Kripke variants, see [17] for an overview.

▶ **Conjecture 59**. FVAL *restricted to dyadic formulas over the* $(\forall, \to)$*-fragment is undecidable.*

The dependence on LEM in Theorem 42 could potentially be eliminated by performing yet another double negation translation. Apart from this, no axioms are assumed.

Reducing to the particular minimal variants becomes feasible because the source problem UDPC is very easy to axiomatize in first-order logic. We consider UDPC a contribution, since it seems to be a useful source problem for compact undecidability reductions in general.

6.2 Comparison to Existing Work

Our results describe minimal undecidable fragments for formulas along two axes: by minimizing the signature, and by minimizing the logical fragment.

A third axis often considered in classical literature is the quantifier prefix. Already in his 1937 work [20], Kalmár not only proves the result about the minimal signature, but also minimizes the quantifier prefix of a formula in prenex normal form to Σ_4. Later work, e.g. by Kalmár [21] and others, strengthened this result. We do not consider minimizing the quantifier prefix, it remains open for future work one might consider.

Results not using Coq work in a classical meta-theory (denoted by FOL_c, which notably does not include intuitionistic PRV), where PRV_c, VAL, and SAT coincide. Further, some historical papers mentioned below presuppose the general undecidability of FOL or otherwise construct a reduction without explicitly aiming to prove any undecidability results.

All cited mechanized results are in Coq. As far as we are aware, there are no mechanizations of first-order undecidability in other proof assistants.

The main difference between our approach and other mechanized results in the literature is that we directly and compactly reduce into dyadic first-order logic, whereas others either need signature compression steps, or otherwise achieve this result in a less direct way (e.g.

Kirst and Hermes [22], who first mechanize meta-theory of ZF set theory). The price for this compactness is paid by starting at a Diophantine constraints-related problem. The undecidability of Diophantine constraints satisfiability was only shown in 1970 by Matiyasevich [27], building on work by Davis, Putnam, and Robinson [6]. Other mechanizations usually start at the Post correspondence problem (PCP) [29], whose undecidability is easier to prove.

Paper	Problems	Dyadic	Small Fragment	In Coq
Church 1936 [4], Turing 1936 [34]	FOL_c	×	×	×
Kalmár 1937 [20]	FOL_c	✓	×	×
Gentzen 1936, Gödel 1929 [13, 14]	FOL	×	$(\forall, \wedge, \neg)$	×
Forster et al. 2019 [9]	FOL, KFOL	×	$(\forall, \to)$	✓
Kirst and Hermes 2021 [22]	FOL	✓	×	✓
The present work	FOL, KFOL	✓	$(\forall, \to)$	✓
Trakhtenbrot 1950 [33]	FSAT	×	×	×
Kirst and Larchey-Wendling 2020 [23]	FSAT	✓	×	✓
The present work	FFOL	✓	$(\forall, \to, \bot)$	✓

6.3 Remarks on the Coq Mechanization

For the mechanization, we start with the PRV reduction outlined in Section 4.3. This is because showing PRV undecidable mostly suffices to show VAL, SAT, and the Kripke variants undecidable, as using soundness subsumes an explicit proof in an abstract model. Thus, Proposition 26 and Proposition 27 are mechanized for provability and not for validity, although the proof is similar. Also, we immediately start in the $(\forall, \to)$-fragment, since the undecidability results for this small fragment subsume those for the large fragment. A separate mechanization of Theorem 29 is included for comprehensibility.

Note that the notation used in Coq differs from the one presented in this paper. In particular, the problem UDPC is `H10UPC`, UDC is `H10UC` similarly. The defining relation $\wr$ of UDPC is called `h10upc_sem_direct`. For $\wr\!\!\wr$, we used `Pr` and sometimes `#` or `##`. For a complete symbol mapping, see the notation mappings at the start of each Coq file.

Following previous projects, we use de Brujin indices [7] to represent binding, which make formulas hard to read and write, especially when there are many nested quantifiers involved. Yet, this made it possible to encode the constraints of an UDPC instance into a first-order formula without having to worry about variable shadowing, as UDPC is also defined with natural numbers as variables in Coq. Thus, our reduction works by ensuring that there are as many free variables as the highest used index in the UDPC source instance.

Hostert et al. [19] developed tools aimed at simplifying reasoning with both de Bruijn indices and the abstract provability predicate. However, we found these to not be applicable, most importantly because our formulas and syntactic proofs feature unboundedly nested quantifiers and an unbound number of hypotheses.

On paper, we have different definitions of $\wr$, and switch between them freely. In Coq, we mostly use Definition 9, as it allows using `lia`, a tactic which automatically solves simple linear integer arithmetic goals. Use of `lia` could be avoided by using the inductive characterization more often. The only place where this characterization makes the proof simpler is Proposition 27.

Our mechanization of the undecidability of PRV and corollaries for the minimal case requires 1000 LoC, which improves upon the approx. 4000 LoC used by Kirst and Hermes [22] for a similar result. Our mechanization for FSAT takes 1200 lines. This also improves upon the results of Kirst and Larchey-Wendling [23], who required about 5000 LoC. In general,

this makes the mechanization of these results in the library much more approachable, as they can be read without following a long reduction chain. The less complicated reduction of Theorem 29, which just minimizes the signature while working within the large logical fragment, takes less than 300 LoC.

We already mentioned Diophantine constraints satisfiability as a large dependency of our mechanization. For comparison, the mechanization used in the library, due to Larchey-Wendling and Forster [25], is 12,000 lines long, and relies on the undecidability of PCP.

References

1 Wilhelm Ackermann. Beiträge zum Entscheidungsproblem der Mathematischen Logik. *Mathematische Annalen*, 112:419–432, 1936. doi:10.1007/BF01565424.

2 Andrej Bauer. First Steps in Synthetic Computability Theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006. Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI). doi:10.1016/j.entcs.2005.11.049.

3 Paul Bernays. Beiträge Zur Reduktionstheorie des Logischen Entscheidungsproblems. *Journal of Symbolic Logic*, 2(2):84–85, 1937. doi:10.2307/2267374.

4 Alonzo Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936. doi:10.2307/2269326.

5 Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76(2):95–120, 1988. doi:10.1016/0890-5401(88)90005-3.

6 Martin Davis, Hilary Putnam, and Julia Robinson. The Decision Problem for Exponential Diophantine Equations. *Annals of Mathematics*, 74(3):425–436, 1961. doi:10.2307/1970289.

7 Nicolaas Govert De Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. In *Indagationes Mathematicae (Proceedings)*, volume 75, pages 381–392. Elsevier, 1972.

8 Yannick Forster. *Computability in Constructive Type Theory*. PhD thesis, Saarland University, 2021. URL: https://ps.uni-saarland.de/~forster/thesis.php.

9 Yannick Forster, Dominik Kirst, and Gert Smolka. 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, pages 38–51, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293880.3294091.

10 Yannick Forster, Dominik Kirst, and Dominik Wehr. Completeness Theorems for First-Order Logic Analysed in Constructive Type Theory: Extended Version. *Journal of Logic and Computation*, 31(1):112–151, January 2021. doi:10.1093/logcom/exaa073.

11 Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq library of undecidable problems. In *CoqPL 2020 The Sixth International Workshop on Coq for Programming Languages*, 2020.

12 Harvey Friedman. Classically and intuitionistically provably recursive functions. In Gert H. Müller and Dana S. Scott, editors, *Higher Set Theory*, pages 21–27, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg. doi:10.1007/BFb0103100.

13 Gerhard Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112:493–565, 1936. doi:10.1007/BF01565428.

14 Kurt Gödel. Zur intuitionistischen Arithmetik und Zahlentheorie – Ergebnisse eines Mathematischen Kolloquiums, 1928-1933.

15 Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37(1):349–360, December 1930. doi:10.1007/BF01696781.

16 Yuri Gurevich. The decision problem for the logic of predicates and of operations. *Algebra and Logic*, 8:160–174, May 1969. doi:10.1007/BF02306690.

17 Hugo Herbelin and Gyesik Lee. Formalizing Logical Metatheory – Semantical Cut-Elimination using Kripke Models for first-order Predicate Logic, 2014. URL: `https://formal.hknu.ac.kr/Kripke/`.

18 David Hilbert and Wilhelm Ackermann. *Grundzüge der Theoretischen Logik*. Springer Verlag, 1928. `doi:10.1007/978-3-662-00049-6`.

19 Johannes Hostert, Mark Koch, and Dominik Kirst. A Toolbox for Mechanised First-Order Logic. In *The Coq Workshop 2021*, 2021.

20 László Kalmár. Zurückführung des Entscheidungsproblems auf den Fall von Formeln mit einer einzigen, binären, Funktionsvariablen. *Compositio Mathematica*, 4:137–144, 1937. URL: `http://www.numdam.org/item/CM_1937__4__137_0/`.

21 László Kalmár. On the reduction of the decision problem. First paper. Ackermann prefix, a single binary predicate. *Journal of Symbolic Logic*, 4(1):1–9, 1939. `doi:10.2307/2266211`.

22 Dominik Kirst and Marc Hermes. 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, 2021. `doi:10.4230/LIPIcs.ITP.2021.23`.

23 Dominik Kirst and Dominique Larchey-Wendling. Trakhtenbrot's Theorem in Coq. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning*, pages 79–96, Cham, 2020. Springer International Publishing. `doi:10.1007/978-3-030-51054-1_5`.

24 Georg Kreisel. On weak completeness of intuitionistic predicate logic. *Journal of Symbolic Logic*, 27(2):139–158, 1962. `doi:10.2307/2964110`.

25 Dominique Larchey-Wendling and Yannick Forster. Hilbert's Tenth Problem in Coq. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. 4th International Conference on Formal Structures for Computation and Deduction, 2019. `doi:10.4230/LIPIcs.FSCD.2019.27`.

26 Leopold Löwenheim. Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, 76:447–470, 1915. `doi:10.1007/BF01458217`.

27 Yuri V. Matiyasevich. Enumerable sets are Diophantine. *Doklady Akademii Nauk SSSR*, 191:279–282, 1970. `doi:10.1142/9789812564894_0013`.

28 Frank Pfenning and Christine Paulin-Mohring. Inductively defined types in the calculus of constructions. In *International Conference on Mathematical Foundations of Programming Semantics*, pages 209–228. Springer, 1989.

29 Emil L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52(4):264–268, 1946.

30 Fred Richman. Church's thesis without tears. *Journal of Symbolic Logic*, 48(3):797–803, 1983. `doi:10.2307/2273473`.

31 Alfred Tarski. I: A General Method in Proofs of Undecidability. In Alfred Tarski, editor, *Undecidable Theories*, volume 13 of *Studies in Logic and the Foundations of Mathematics*, pages 1–34. Elsevier, 1953. `doi:10.1016/S0049-237X(09)70292-7`.

32 The Coq Development Team. The Coq Proof Assistant, January 2021. `doi:10.5281/zenodo.4501022`.

33 Boris Trakhtenbrot. The Impossibility of an Algorithm for the Decidability Problem on Finite Classes. In *Proceedings of the USSR Academy of Sciences*, 1950.

34 Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. `doi:10.1112/plms/s2-42.1.230`.