


# Trakhtenbrot’s Theorem in Coq

## A Constructive Approach to Finite Model Theory

Dominik Kirst<sup>1</sup> , Dominique Larchey-Wendling<sup>2</sup> 

<sup>1</sup>Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

<sup>2</sup>Université de Lorraine, CNRS, LORIA, Vandœuvre-lès-Nancy, France  
kirst@ps.uni-saarland.de      dominique.larchey-wendling@loria.fr

**Abstract.** We study finite first-order satisfiability (FSAT) in the constructive setting of dependent type theory. Employing synthetic accounts of enumerability and decidability, we give a full classification of FSAT depending on the first-order signature of non-logical symbols. On the one hand, our development focuses on Trakhtenbrot’s theorem, stating that FSAT is undecidable as soon as the signature contains an at least binary relation symbol. Our proof proceeds by a many-one reduction chain starting from the Post correspondence problem. On the other hand, we establish the decidability of FSAT for monadic first-order logic, i.e. where the signature only contains at most unary function and relation symbols, as well as the enumerability of FSAT for arbitrary enumerable signatures. All our results are mechanised in the framework of a growing Coq library of synthetic undecidability proofs.

## 1 Introduction

In the wake of the seminal discoveries concerning the undecidability of first-order logic by Turing and Church in the 1930s, a broad line of work has been pursued to characterise the border between decidable and undecidable fragments of the original decision problem. These fragments can be grouped either by syntactic restrictions controlling the allowed function and relation symbols or the quantifier prefix, or by semantic restrictions on the admitted models (see [1] for a comprehensive description).

Concerning signature restrictions, already predating the undecidability results, Löwenheim had shown in 1915 that monadic first-order logic, admitting only signatures with at most unary symbols, is decidable [15]. Therefore, the successive negative results usually presuppose non-trivial signatures containing an at least binary symbol.

Turning to semantic restrictions, Trakhtenbrot proved in 1950 that, if only admitting finite models, the satisfiability problem over non-trivial signatures is still undecidable [21]. Moreover, the situation is somewhat dual to the unrestricted case, since finite satisfiability (FSAT) is still enumerable while, in the unrestricted case, validity is enumerable. As a consequence, finite validity cannot be characterised by a complete finitary deduction system and, resting on finite model theory, various natural problems in database theory are undecidable.

Conventionally, Trakhtenbrot’s theorem is proved by (many-one) reduction from the halting problem for Turing machines (see e.g. [1,14]). An encoding of a given Turing machine  $M$  can be given as a formula  $\varphi_M$  such that the models of  $\varphi_M$  correspond to the runs of  $M$ . Specifically, the finite models of  $\varphi_M$  correspond to terminating runs of  $M$  and so a decision procedure for finite satisfiability of  $\varphi_M$  would be enough to decide whether  $M$  terminates or not.

Although this proof strategy is in principle explainable on paper, already the formal definition of Turing machines, not to mention their encoding in first-order logic, is not ideal for mechanisation in a proof assistant. So for our Coq mechanisation of Trakhtenbrot’s theorem, we follow a different strategy by starting from the Post correspondence problem (PCP), a simple matching problem on strings. Similar to the conventional proof, we proceed by encoding every instance  $R$  of PCP as a formula  $\varphi_R$  such that  $R$  admits a solution iff  $\varphi_R$  has a finite model. Employing the framework of synthetic undecidability [8,11], the computability of  $\varphi_R$  from  $R$  is guaranteed since all functions definable in constructive type theory are computable without reference to a concrete model of computation.

Both the conventional proof relying on Turing machines and our elaboration starting from PCP actually produce formulas in a custom signature well-suited for the encoding of the seed decision problems. The sharper version of Trakhtenbrot’s theorem, stating that a signature with at least one binary relation (or one binary function and one unary relation) is enough to turn FSAT undecidable, is in fact left as an exercise in e.g. Libkin’s book [14]. However, at least in a constructive setting, this generalisation is non-trivial and led us to mechanising a chain of signature transformations eliminating and compressing function and relation symbols step by step.

Complementing the undecidability result, we further formalise that FSAT is enumerable for enumerable signatures and decidable for monadic signatures. Again, both of these standard results come with their subtleties when explored in a constructive approach of finite model theory.

In summary, the main contributions of this paper are threefold:

- we provide an axiom-free Coq mechanisation comprising a full classification of finite satisfiability with regards to the signatures allowed;<sup>1</sup>
- we present a streamlined proof strategy for Trakhtenbrot’s theorem well-suited for mechanisation and simple to explain informally, basing on PCP;
- we give a constructive account of signature transformations and the treatment of interpreted equality typically neglected in a classical development.

The rest of the paper is structured as follows. We first describe the type-theoretical framework for undecidability proofs and the representation of first-order logic in Section 2. We then outline our variant of Trakhtenbrot’s theorem for a custom signature in Section 3. This is followed by a development of enough constructive finite model theory (Section 4) to conclude some decidability results (Section 5) as well as the final classification (Section 6). We end with a brief discussion of the Coq development and future work in Section 7.

<sup>1</sup> Downloadable from <http://www.ps.uni-saarland.de/extras/fo1-trakh/> and systematically hyperlinked with the definitions and theorems in this PDF.

## 2 First-Order Satisfiability in Constructive Type Theory

In order to make this paper accessible to readers unfamiliar with constructive type theory, we outline the required features of Coq’s underlying type theory, the synthetic treatment of computability available in constructive mathematics, some properties of finite types, as well as our representation of first-order logic.

### 2.1 Basics of Constructive Type Theory

We work in the framework of a constructive type theory such as the one implemented in Coq, providing a predicative hierarchy of type universes  $\mathbb{T}$  above a single impredicative universe  $\mathbb{P}$  of propositions. On type level, we have the unit type  $\mathbb{1}$  with a single element  $*$  :  $\mathbb{1}$ , the void type  $\mathbb{0}$ , function spaces  $X \rightarrow Y$ , products  $X \times Y$ , sums  $X + Y$ , dependent products  $\forall x : X. F x$ , and dependent sums  $\{x : X \mid F x\}$ . On propositional level, these types are denoted using the usual logical notation ( $\top$ ,  $\perp$ ,  $\rightarrow$ ,  $\wedge$ ,  $\vee$ ,  $\forall$ , and  $\exists$ ).

We employ the basic inductive types of Booleans ( $\mathbb{B} ::= \text{tt} \mid \text{ff}$ ), of Peano natural numbers ( $n : \mathbb{N} ::= 0 \mid S n$ ), the option type ( $\mathbb{O} X ::= \ulcorner x \urcorner \mid \emptyset$ ), and lists ( $l : \mathbb{L} X ::= [] \mid x :: l$ ). We write  $|l|$  for the *length* of a list,  $l \# m$  for the *concatenation* of  $l$  and  $m$ ,  $x \in l$  for *membership*, and simply  $f [x_1; \dots; x_n] := [f x_1; \dots; f x_n]$  for the *map* function. We denote by  $X^n$  the type of vectors of length  $n : \mathbb{N}$  and by  $\mathbb{F}_n$  the finite types understood as indices  $\{0, \dots, n - 1\}$ . The definitions/notations for lists are shared with vectors  $\mathbf{v} : X^n$ . Moreover, when  $i : \mathbb{F}_n$  and  $x : X$ , we denote by  $\mathbf{v}_i$  the  $i$ -th component of  $\mathbf{v}$  and by  $\mathbf{v}[x/i]$  the vector  $\mathbf{v}$  with  $i$ -th component updated to value  $x$ .

### 2.2 Synthetic (Un-)decidability

We review the main ingredients of our synthetic approach to decidability and undecidability [7,8,10,11,13,19], based on the computability of all functions definable in constructive type theory.<sup>2</sup> We first introduce standard notions of computability theory without referring to a formal model of computation, e.g. Turing machines.

**Definition 1.** A *problem* or predicate  $p : X \rightarrow \mathbb{P}$  is

- *decidable* if there is  $f : X \rightarrow \mathbb{B}$  with  $\forall x. p x \leftrightarrow f x = \text{tt}$ .
- *enumerable* if there is  $f : \mathbb{N} \rightarrow \mathbb{O} X$  with  $\forall x. p x \leftrightarrow \exists n. f n = \ulcorner x \urcorner$ .

These notions generalise to predicates of higher arity. Moreover, a type  $X$  is

- *enumerable* if there is  $f : \mathbb{N} \rightarrow \mathbb{O} X$  with  $\forall x. \exists n. f n = \ulcorner x \urcorner$ .
- *discrete* if equality on  $X$  (i.e.  $\lambda xy : X. x = y$ ) is decidable.
- a *data type* if it is both enumerable and discrete.

<sup>2</sup> A result shown and applied for many variants of constructive type theory and which Coq designers are committed to maintain as Coq evolves.

Using the expressiveness of dependent types, we equivalently tend to establish the decidability of a predicate  $p : X \rightarrow \mathbb{P}$  by giving a function  $\forall x : X. px + \neg px$ . Note that it is common to mechanise decidability results in this synthetic sense (e.g. [2,16,17]). Next, decidability and enumerability transport along reductions:

**Definition 2.** A problem  $p : X \rightarrow \mathbb{P}$  (*many-one*) *reduces* to  $q : Y \rightarrow \mathbb{P}$ , written  $p \preceq q$ , if there is a function  $f : X \rightarrow Y$  such that  $px \leftrightarrow q(fx)$  for all  $x : X$ .<sup>3</sup>

**Fact 3.** *Assume  $p : X \rightarrow \mathbb{P}$ ,  $q : Y \rightarrow \mathbb{P}$  and  $p \preceq q$ : (1) if  $q$  is decidable, then so is  $p$  and (2) if  $X$  and  $Y$  are data types and  $q$  is enumerable, then so is  $p$ .*

Item (1) implies that we can justify the undecidability of a target problem by reduction from a seed problem known to be undecidable, such as the halting problem for Turing machines. This is in fact the closest rendering of undecidability available in a synthetic setting, since the underlying type theory is consistent with the assumption that every problem is decidable.<sup>4</sup> Nevertheless, we believe that in the intended effective interpretation for synthetic computability, a typical seed problem is indeed undecidable and so are the problems reached by verified reductions.<sup>5</sup> More specifically, since the usual seed problems are not co-enumerable, (2) implies that the reached problems are not co-enumerable either.

Given its simple inductive characterisation involving only basic types of lists and Booleans, the (binary) Post correspondence problem (BPCP) is a well-suited seed problem for compact encoding into first-order logic.

**Definition 4.** Given a list  $R : \mathbb{L}(\mathbb{L}\mathbb{B} \times \mathbb{L}\mathbb{B})$  of pairs  $s/t$  of Boolean strings,<sup>6</sup> we define *derivability* of a pair  $s/t$  from  $R$  (denoted by  $R \triangleright s/t$ ) and *solvability* (denoted by  $\text{BPCP } R$ ) by the following rules:

$$\frac{s/t \in R}{R \triangleright s/t} \qquad \frac{s/t \in R \quad R \triangleright u/v}{R \triangleright (s \# u)/(t \# v)} \qquad \frac{R \triangleright s/s}{\text{BPCP } R}$$

**Fact 5.** *Given a list  $R : \mathbb{L}(\mathbb{L}\mathbb{B} \times \mathbb{L}\mathbb{B})$ , the derivability predicate  $\lambda st. R \triangleright s/t$  is decidable. However, the halting problem for Turing machines reduces to BPCP.*

*Proof.* We give of proof of the decidability of  $R \triangleright s/t$  by induction on  $|s| + |t|$ . We also provide a trivial proof of the equivalence of two definitions of BPCP. See [7,10] for details on the reduction from the halting problem to BPCP.  $\square$

It might at first appear surprising that derivability  $\lambda st. R \triangleright s/t$  is decidable while BPCP is reducible from the halting problem (and hence undecidable). This simply illustrates that undecidability is caused by the unbounded existential quantifier in the equivalence  $\text{BPCP } R \leftrightarrow \exists s. R \triangleright s/s$ .

<sup>3</sup> Or equivalently, the dependent characterisation  $\forall x : X. \{y : Y \mid px \leftrightarrow qy\}$ .

<sup>4</sup> As witnessed by classical set-theoretic models satisfying  $\forall p : \mathbb{P}. p + \neg p$  (cf. [23]).

<sup>5</sup> This synthetic treatment of undecidability is discussed in more detail in [8] and [11].

<sup>6</sup> Notice that the list  $R$  is viewed as a (finite) set of pairs  $s/t \in R$  (hence ignoring the order or duplicates), while  $s$  and  $t$ , which are also lists, are viewed as strings (hence repetitions and ordering matter for  $s$  and  $t$ ).

### 2.3 Finiteness

**Definition 6.** A type  $X$  is *finite* if there is a list  $l_X$  with  $x \in l_X$  for all  $x : X$  and a *predicate*  $p : X \rightarrow \mathbb{P}$  is *finite* if there is a list  $l_p$  with  $\forall x. px \leftrightarrow x \in l_p$ .

Note that in constructive settings there are various alternative characterisations of finiteness<sup>7</sup> (bijection with  $\mathbb{F}_n$  for some  $n$ ; negated infinitude for some definition of infiniteness; etc.) and we opted for the above since it is easy to work with while transparently capturing the expected meaning. One can distinguish *strong* finiteness in  $\mathbb{T}$  (i.e.  $\{l_X : \mathbb{L}X \mid \forall x. x \in l_X\}$ ) from *weak* finiteness in  $\mathbb{P}$  (i.e.  $\exists l_X : \mathbb{L}X. \forall x. x \in l_X$ ), the list  $l_X$  being required computable in the strong case.

We present three important tools for manipulating finite types: the *finite pigeon hole principle* (PHP) here established without assuming discreteness, the *well-foundedness* of strict orders over finite types, and *quotients* over strongly decidable equivalences that map onto  $\mathbb{F}_n$ . The proofs are given in Appendix A of the extended version of this paper [12].

For the finite PHP, the typical classical proof requires the discreteness of  $X$  to design transpositions/permutations. Here we avoid discreteness completely, the existence of a duplicate being established without actually computing one.

**Theorem 7. (Finite PHP)** *Let  $R : X \rightarrow Y \rightarrow \mathbb{P}$  be a binary relation and  $l : \mathbb{L}X$  and  $m : \mathbb{L}Y$  be two lists where  $m$  is shorter than  $l$  ( $|m| < |l|$ ). If  $R$  is total from  $l$  to  $m$  ( $\forall x. x \in l \rightarrow \exists y. y \in m \wedge Rx y$ ) then the values at two distinct positions in  $l$  are related to the same  $y$  in  $m$ , i.e. there exist  $x_1, x_2 \in l$  and  $y \in m$  such that  $l$  has shape  $l = \dots \# x_1 :: \dots \# x_2 :: \dots$  and  $Rx_1 y$  and  $Rx_2 y$ .*

Using the PHP, one can constructively show that, for a strict order over a finite type  $X$ , any descending chain has length bounded by the size of  $X$ .<sup>8</sup>

**Fact 8.** *Every strict order on a finite type is well-founded.*

Coq's type theory does not provide quotients in general (see e.g. [6]) but one can build computable quotients in certain conditions, here for a decidable equivalence relation of which representatives of equivalence classes are listable.

**Theorem 9. (Finite decidable quotient)** *Let  $\sim : X \rightarrow X \rightarrow \mathbb{P}$  be a decidable equivalence with  $\{l_r : \mathbb{L}X \mid \forall x \exists y. y \in l_r \wedge x \sim y\}$ , i.e. finitely many equivalence classes.<sup>9</sup> Then one can compute the quotient  $X/\sim$  onto  $\mathbb{F}_n$  for some  $n$ , i.e.  $n : \mathbb{N}$ ,  $c : X \rightarrow \mathbb{F}_n$  and  $r : \mathbb{F}_n \rightarrow X$  s.t.  $\forall p. c(rp) = p$  and  $\forall x y. x \sim y \leftrightarrow cx = cy$ .*

Using Theorem 9 with identity over  $X$  as equivalence, we get bijections between finite, discrete types and the type family  $(\mathbb{F}_n)_{n:\mathbb{N}}$ .<sup>10</sup>

**Corollary 10.** *If  $X$  is a finite and discrete type then one can compute  $n : \mathbb{N}$  and a bijection from  $X$  to  $\mathbb{F}_n$ .*

<sup>7</sup> And these alternative characterisations are not necessarily constructively equivalent.

<sup>8</sup> i.e. the length of the enumerating list of  $X$ .

<sup>9</sup> Hence  $l_r$  denotes a list of *representatives* of equivalence classes.

<sup>10</sup> For a given  $X$ , the value  $n$  (usually called cardinal) is unique by the PHP.

## 2.4 Representing First-Order Logic

We briefly outline our representation of the syntax and semantics of first-order logic in constructive type theory (cf. [9]). Concerning the *syntax*, we describe terms and formulas as dependent inductive types over 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  $|P|$ , using binary connectives  $\Box \in \{\rightarrow, \wedge, \dot{\vee}\}$  and quantifiers  $\nabla \in \{\dot{\forall}, \dot{\exists}\}$ :

$$\begin{aligned} t : \mathbf{Term}_\Sigma &::= x \mid f \mathbf{t} & (x : \mathbb{N}, f : \mathcal{F}_\Sigma, \mathbf{t} : \mathbf{Term}_\Sigma^{|f|}) \\ \varphi, \psi : \mathbf{Form}_\Sigma &::= \perp \mid P \mathbf{t} \mid \varphi \Box \psi \mid \nabla \varphi & (P : \mathcal{P}_\Sigma, \mathbf{t} : \mathbf{Term}_\Sigma^{|P|}) \end{aligned}$$

Negation is defined as the abbreviation  $\dot{\neg}\varphi := \varphi \rightarrow \perp$ .

In the chosen de Bruijn representation [4], a bound variable is encoded as the number of quantifiers shadowing its binder, e.g.  $\forall x. \exists y. P x u \rightarrow P y v$  may be represented by  $\dot{\forall} \dot{\exists} P 1 4 \rightarrow P 0 5$ . The variables  $2 = 4 - 2$  and  $3 = 5 - 2$  in this example are the *free* variables, and variables that do not occur freely are called *fresh*, e.g. 0 and 1 are fresh. For the sake of legibility, we write concrete formulas with named binders and defer de Bruijn representations to the Coq development. For a formula  $\varphi$  over a signature  $\Sigma$ , we define the list  $\mathbf{FV}(\varphi) : \mathbb{L} \mathbb{N}$  of free variables, the list  $\mathcal{F}_\varphi : \mathbb{L} \mathcal{F}_\Sigma$  of function symbols and the list  $\mathcal{P}_\varphi : \mathbb{L} \mathcal{P}_\Sigma$  of relation symbols that actually occur in  $\varphi$ , all by recursion on  $\varphi$ .

Turning to *semantics*, we employ the standard (Tarski-style) model-theoretic semantics, evaluating terms in a given domain and embedding the logical connectives into the constructive meta-logic (cf. [22]):

**Definition 11.** A *model*  $\mathcal{M}$  over a domain  $D : \mathbb{T}$  is described by a pair of functions  $\forall f. D^{|f|} \rightarrow D$  and  $\forall P. D^{|P|} \rightarrow \mathbb{P}$  denoted by  $f^\mathcal{M}$  and  $P^\mathcal{M}$ . Given a *variable assignment*  $\rho : \mathbb{N} \rightarrow D$ , we recursively extend it to a *term evaluation*  $\hat{\rho} : \mathbf{Term} \rightarrow D$  with  $\hat{\rho} x := \rho x$  and  $\hat{\rho}(f \mathbf{v}) := f^\mathcal{M}(\hat{\rho} \mathbf{v})$ , and to the *satisfaction* relation  $\mathcal{M} \models_\rho \varphi$  by

$$\begin{aligned} \mathcal{M} \models_\rho \perp &:= \perp & \mathcal{M} \models_\rho \varphi \Box \psi &:= \mathcal{M} \models_\rho \varphi \Box \mathcal{M} \models_\rho \psi \\ \mathcal{M} \models_\rho P \mathbf{t} &:= P^\mathcal{M}(\hat{\rho} \mathbf{t}) & \mathcal{M} \models_\rho \nabla \varphi &:= \nabla a : D. \mathcal{M} \models_{a \cdot \rho} \varphi \end{aligned}$$

where each logical connective  $\Box/\nabla$  is mapped to its meta-level counterpart  $\Box/\nabla$  and where we denote by  $a \cdot \rho$  the de Bruijn extension of  $\rho$  by  $a$ , defined by  $(a \cdot \rho) 0 := a$  and  $(a \cdot \rho)(1 + x) := \rho x$ .<sup>11</sup>

A  $\Sigma$ -*model* is thus a dependent triple  $(D, \mathcal{M}, \rho)$  composed of a domain  $D$ , a model  $\mathcal{M}$  for  $\Sigma$  over  $D$  and an assignment  $\rho : \mathbb{N} \rightarrow D$ . It is *finite* if  $D$  is finite, and *decidable* if  $P^\mathcal{M} : D^{|P|} \rightarrow \mathbb{P}$  is decidable for all  $P : \mathcal{P}_\Sigma$ .

**Fact 12.** *Satisfaction*  $\lambda \varphi. \mathcal{M} \models_\rho \varphi$  is decidable for finite, decidable  $\Sigma$ -models.

*Proof.* By induction on  $\varphi$ ; finite quantification preserves decidability.  $\square$

<sup>11</sup> The notation  $a \cdot \rho$  illustrates that  $a$  is pushed ahead of the sequence  $\rho_0, \rho_1, \dots$

In this paper, we are mostly concerned with *finite satisfiability* of formulas. However, since some of the compound reductions hold for more general or more specific notions, we introduce the following variants:

**Definition 13. (Satisfiability)** For a formula  $\varphi$  over a signature  $\Sigma$ , we write

- $\text{SAT}(\Sigma) \varphi$  if there is a  $\Sigma$ -model  $(D, \mathcal{M}, \rho)$  such that  $\mathcal{M} \models_\rho \varphi$ ;
- $\text{FSAT}(\Sigma) \varphi$  if additionally  $D$  is finite and  $\mathcal{M}$  is decidable;
- $\text{FSATEQ}(\Sigma; \equiv) \varphi$  if the signature contains a distinguished binary relation symbol  $\equiv$  interpreted as equality, i.e.  $x \equiv^{\mathcal{M}} y \leftrightarrow x = y$  for all  $x, y : D$ .

Notice that in a classical treatment of finite model theory, models are supposed to be given *in extension*, i.e. understood as tables providing computational access to functions and relations values. To enable this view in our constructive setting, we restrict to decidable relations in the definition of FSAT, and from now on, *finite satisfiability is always meant to encompass a decidable model*. One could further require the domain  $D$  to be discrete to conform more closely with the classical view; discreteness is in fact enforced by FSATEQ. However, we refrain from this requirement and instead show in Section 4.1 that FSAT and FSAT over discrete models are constructively equivalent.

### 3 Trakhtenbrot's Theorem for a Custom Signature

In this section, we show that BPCP reduces to  $\text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv)$  for the special purpose signature  $\Sigma_{\text{BPCP}} := (\{\star^0, e^0, f_{\text{tt}}^1, f_{\text{ff}}^1\}; \{P^2, \prec^2, \equiv^2\})$ . To this end, we fix an instance  $R : \mathbb{L}(\mathbb{L}\mathbb{B} \times \mathbb{L}\mathbb{B})$  of BPCP (to be understood as a finite set of pairs of Boolean strings) and we construct a formula  $\varphi_R$  such that  $\varphi_R$  is finitely satisfiable if and only if  $R$  has a solution.

Informally, we axiomatise a family  $\mathcal{B}_n$  of models over the domain of Boolean strings of length bounded by  $n$  and let  $\varphi_R$  express that  $R$  has a solution in  $\mathcal{B}_n$ . The axioms express enough equations and inversions of the constructions included in the definition of BPCP such that a solution for  $R$  can be recovered.

Formally, the symbols in  $\Sigma_{\text{BPCP}}$  are used as follows: the functions  $f_b$  and the constant  $e$  represent  $b :: (\cdot)$  and  $[\ ]$  for the encoding of strings  $s$  as terms  $\bar{s}$ :

$$\overline{[\ ]} \# \tau := \tau \quad \overline{b :: s} \# \tau := f_b(\bar{s} \# \tau) \quad \bar{s} := \bar{s} \# e$$

The constant  $\star$  represents an undefined value for strings too long to be encoded in the finite model  $\mathcal{B}_n$ . The relation  $P$  represents derivability from  $R$  (denoted  $R \triangleright \cdot / \cdot$  here) while  $\prec$  and  $\equiv$  represent strict suffixes and equality, respectively.

Expected properties of the intended interpretation can be captured formally as first-order formulas. First, we ensure that  $P$  is proper (only subject to defined values) and that  $\prec$  is a strict order (irreflexive and transitive):

$$\begin{aligned} \varphi_P &:= \dot{\forall}xy. Pxy \dot{\rightarrow} x \not\equiv \star \wedge y \not\equiv \star && (P \text{ proper}) \\ \varphi_{\prec} &:= (\dot{\forall}x. x \not\prec x) \wedge (\dot{\forall}xyz. x \prec y \dot{\rightarrow} y \prec z \dot{\rightarrow} x \prec z) && (\prec \text{ strict order}) \end{aligned}$$

Next, the image of  $f_b$  is forced disjoint from  $e$  and injective as long as  $\star$  is not reached. We also ensure that the images of  $f_{tt}$  and  $f_{ff}$  intersect only at  $\star$ :

$$\varphi_f := \left( \begin{array}{l} f_{tt} \star \equiv \star \wedge f_{ff} \star \equiv \star \\ \dot{\forall} x. f_{tt} x \not\equiv e \\ \dot{\forall} x. f_{ff} x \not\equiv e \end{array} \right) \wedge \left( \begin{array}{l} \dot{\forall} xy. f_{tt} x \not\equiv \star \dot{\rightarrow} f_{tt} x \equiv f_{tt} y \dot{\rightarrow} x \equiv y \\ \dot{\forall} xy. f_{ff} x \not\equiv \star \dot{\rightarrow} f_{ff} x \equiv f_{ff} y \dot{\rightarrow} x \equiv y \\ \dot{\forall} xy. f_{tt} x \equiv f_{ff} y \dot{\rightarrow} f_{tt} x \equiv \star \wedge f_{ff} y \equiv \star \end{array} \right)$$

Furthermore, we enforce that  $P$  simulates  $R \triangleright \cdot / \cdot$ , encoding its inversion principle

$$\varphi_{\triangleright} := \dot{\forall} xy. P x y \dot{\rightarrow} \bigvee_{s/t \in R} \dot{\forall} \left\{ \begin{array}{l} x \equiv \bar{s} \wedge y \equiv \bar{t} \\ \dot{\exists} uv. P u v \wedge x \equiv \bar{s} \# u \wedge y \equiv \bar{t} \# v \wedge u/v \prec x/y \end{array} \right.$$

where  $u/v \prec x/y$  denotes  $(u \prec x \wedge v \equiv y) \dot{\vee} (v \prec y \wedge u \equiv x) \dot{\vee} (u \prec x \wedge v \prec y)$ . Finally,  $\varphi_R$  is the conjunction of all axioms plus the existence of a solution:

$$\varphi_R := \varphi_P \wedge \varphi_{\prec} \wedge \varphi_f \wedge \varphi_{\triangleright} \wedge \dot{\exists} x. P x x.$$

**Theorem 14.**  $\text{BPCP} \preceq \text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv)$ .

*Proof.* The reduction  $\lambda R. \varphi_R$  is proved correct by Lemmas 15 and 16.  $\square$

**Lemma 15.**  $\text{BPCP } R \rightarrow \text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv) \varphi_R$ .

*Proof.* Assume  $R \triangleright s/s$  holds for a string  $s$  with  $|s| = n$ . We show that the model  $\mathcal{B}_n$  over Boolean strings bounded by  $n$  satisfies  $\varphi_R$ . To be more precise, we choose  $D_n := \mathbb{O}\{s : \mathbb{L}\mathbb{B} \mid |s| \leq n\}$  as domain, i.e. values in  $D_n$  are either an (overflow) value  $\emptyset$  or a (defined) dependent pair  $\ulcorner (s, H_s) \urcorner$  where  $H_s : |s| \leq n$ . We interpret the function and relation symbols of the chosen signature by

$$\begin{array}{ll} e^{\mathcal{B}_n} := [] & f_b^{\mathcal{B}_n} \emptyset := \emptyset \\ \star^{\mathcal{B}_n} := \emptyset & f_b^{\mathcal{B}_n} s := \text{if } |s| < n \text{ then } b :: s \text{ else } \emptyset \end{array} \quad \begin{array}{l} P^{\mathcal{B}_n} s t := R \triangleright s/t \\ s \prec^{\mathcal{B}_n} t := s \neq t \wedge \exists u. u \# s = t \end{array}$$

where we left out some explicit constructors and the excluded edge cases of the relations for better readability. As required,  $\mathcal{B}_n$  interprets  $\equiv$  by equality  $=_{D_n}$ .

Considering the desired properties of  $\mathcal{B}_n$ , first note that  $D_n$  can be shown finite by induction on  $n$ . This however crucially relies on the proof irrelevance of the  $\lambda x. x \leq n$  predicate!<sup>12</sup> The atoms  $s \prec^{\mathcal{B}_n} t$  and  $s \equiv^{\mathcal{B}_n} t$  are decidable by straightforward computations on Boolean strings. Decidability of  $P^{\mathcal{B}_n} s t$  (i.e.  $R \triangleright s/t$ ) was established in Fact 5. Finally, since  $\varphi_R$  is a closed formula, any variable assignment  $\rho$  can be chosen to establish that  $\mathcal{B}_n$  satisfies  $\varphi_R$ , for instance  $\rho := \lambda x. \emptyset$ . Then showing  $\mathcal{B}_n \models_{\rho} \varphi_R$  consists of verifying simple properties of the chosen functions and relations, with mostly straightforward proofs.  $\square$

**Lemma 16.**  $\text{FSATEQ}(\Sigma_{\text{BPCP}}; \equiv) \varphi_R \rightarrow \text{BPCP } R$ .

<sup>12</sup> i.e. that for every  $x : \mathbb{N}$  and  $H, H' : x \leq n$  we have  $H = H'$ . In general, it is not always possible to establish finiteness of  $\{x \mid P x\}$  if  $P$  is not proof irrelevant.



*Proof.* Suppose that  $\mathcal{M} \models_{\rho} \varphi_R$  holds for some finite  $\Sigma_{\text{BPCP}}$ -model  $(D, \mathcal{M}, \rho)$  interpreting  $\equiv$  as equality and providing operations  $f_b^{\mathcal{M}}, e^{\mathcal{M}}, \star^{\mathcal{M}}, P^{\mathcal{M}}$  and  $\prec^{\mathcal{M}}$ . Again, the concrete assignment  $\rho$  is irrelevant and  $\mathcal{M} \models_{\rho} \varphi_R$  ensures that the functions/relations behave as specified and that  $P^{\mathcal{M}} x x$  holds for some  $x : D$ .

Instead of trying to show that  $\mathcal{M}$  is isomorphic to some  $\mathcal{B}_n$ , we directly reconstruct a solution for  $R$ , i.e. we find some  $s$  with  $R \triangleright s/s$  from the assumption that  $\mathcal{M} \models_{\rho} \varphi_R$  holds. To this end, we first observe that the relation  $u/v \prec^{\mathcal{M}} x/y$  as defined above is a strict order and thus well-founded as an instance of Fact 8.

Now we can show that for all  $x/y$  with  $P^{\mathcal{M}} x y$  there are strings  $s$  and  $t$  with  $x = \bar{s}, y = \bar{t}$  and  $R \triangleright s/t$ , by induction on the pair  $x/y$  using the well-foundedness of  $\prec^{\mathcal{M}}$ . So let us assume  $P^{\mathcal{M}} x y$ . Since  $\mathcal{M}$  satisfies  $\varphi_{\triangleright}$  there are two cases:

- there is  $s/t \in R$  such that  $x = \bar{s}$  and  $y = \bar{t}$ . The claim follows by  $R \triangleright s/t$ ;
- there are  $u, v : D$  with  $P^{\mathcal{M}} u v$  and  $s/t \in R$  such that  $x = \bar{s} \# u, y = \bar{t} \# v$ , and  $u/v \prec^{\mathcal{M}} x/y$ . The latter makes the inductive hypothesis applicable for  $P^{\mathcal{M}} u v$ , hence yielding  $R \triangleright s'/t'$  for some strings  $s'$  and  $t'$  corresponding to the encodings  $u$  and  $v$ . This is enough to conclude  $x = \bar{s} \# s', y = \bar{t} \# t'$  and  $R \triangleright (s \# s')/(t \# t')$  as wished.

Applying this fact to the assumed match  $P^{\mathcal{M}} x x$  yields a solution  $R \triangleright s/s$ .  $\square$

## 4 Constructive Finite Model Theory

Combined with Fact 5, Theorem 14 entails the undecidability (and non-co-enumerability) of FSATEQ over a custom (both finite and discrete) signature  $\Sigma_{\text{BPCP}}$ . By a series of signature reductions, we generalise these results to any signature containing an at least binary relation symbol. In particular, we explain how to reduce  $\text{FSAT}(\Sigma)$  to  $\text{FSAT}(\emptyset; \{\epsilon^2\})$  for any discrete signature  $\Sigma$ , hence including  $\Sigma_{\text{BPCP}}$ . We also provide a reduction from  $\text{FSAT}(\emptyset; \{\epsilon^2\})$  to  $\text{FSAT}(\{f^n\}; \{P^1\})$  for  $n \geq 2$ , which entails the undecidability of FSAT for signatures with one unary relation and an at least binary function. But first, let us show that FSAT is unaltered when further assuming discreteness of the domain.

### 4.1 Removing Model Discreteness and Interpreted Equality

We consider the case of models over a discrete domain  $D$ . Of course, in the case of  $\text{FSATEQ}(\Sigma; \equiv)$  the requirement that  $\equiv$  is interpreted as a decidable binary relation which is equivalent to  $=_D$  imposes the discreteness of  $D$ . But in the case of  $\text{FSAT}(\Sigma)$  nothing imposes such a restriction on  $D$ . However, as we argue here, we can always quotient  $D$  using a suitable decidable congruence, making the quotient a discrete finite type while preserving first-order satisfaction.

**Definition 17.** We write  $\text{FSAT}'(\Sigma) \varphi$  if  $\text{FSAT}(\Sigma) \varphi$  on a discrete model.

Let us consider a fixed signature  $\Sigma = (\mathcal{F}_{\Sigma}; \mathcal{P}_{\Sigma})$ . In addition, let us fix a finite type  $D$  and a (decidable) model  $\mathcal{M}$  of  $\Sigma$  over  $D$ . We can conceive an equivalence

over  $D$  which is a congruence for all the interpretations of the symbols by  $\mathcal{M}$ , namely *first-order indistinguishability*  $x \dot{=}_{\Sigma} y := \forall \varphi \rho. \mathcal{M} \models_{x.\rho} \varphi \leftrightarrow \mathcal{M} \models_{y.\rho} \varphi$ , i.e. first-order semantics in  $\mathcal{M}$  is not impacted when switching  $x$  with  $y$ .

The facts that  $\dot{=}_{\Sigma}$  is both an equivalence and a congruence are easy to prove but, with this definition, there is little hope of establishing decidability of  $\dot{=}_{\Sigma}$ . The main reason for this is that the signature may contain symbols of infinitely many arities. So we fix two lists  $l_{\mathcal{F}} : \mathbb{L} \mathcal{F}_{\Sigma}$  and  $l_{\mathcal{P}} : \mathbb{L} \mathcal{P}_{\Sigma}$  of function and relation symbols respectively and restrict the congruence requirement to these lists.

**Definition 18. (Bounded first-order indistinguishability)** We say that  $x$  and  $y$  are *first-order indistinguishable up to  $l_{\mathcal{F}}/l_{\mathcal{P}}$* , and we write  $x \dot{=} y$ , if for any  $\rho : \mathbb{N} \rightarrow D$  and any first-order formula  $\varphi$  built from the symbols in  $l_{\mathcal{F}}$  and  $l_{\mathcal{P}}$  only, we have  $\mathcal{M} \models_{x.\rho} \varphi \leftrightarrow \mathcal{M} \models_{y.\rho} \varphi$ .

**Theorem 19.** *First-order indistinguishability  $\dot{=}$  up to  $l_{\mathcal{F}}/l_{\mathcal{P}}$  is a strongly decidable equivalence and a congruence for all the symbols in  $l_{\mathcal{F}}/l_{\mathcal{P}}$ .*

*Proof.* The proof is quite involved, we only give its sketch here; see Appendix B of the extended version of this paper [12] for more details. The real difficulty is to show the decidability of  $\dot{=}$ . To this end, we characterise  $\dot{=}$  as a bisimulation, i.e. we show that  $\dot{=}$  is extensionally equivalent to Kleene's greatest fixpoint  $F^{\omega}(\lambda uv. \top)$  of some  $\omega$ -continuous operator  $F : (D \rightarrow D \rightarrow \mathbb{P}) \rightarrow (D \rightarrow D \rightarrow \mathbb{P})$ . We then show that  $F$  preserves strong decidability. To be able to conclude, we establish that  $F$  reaches its limit after  $l := 2^{d \times d}$  iterations where  $d := \text{card } D$ , the length of a list enumerating the finite type  $D$ . To verify this upper bound, we build the *weak powerset*, a list of length  $l$  which contains all the weakly decidable binary predicates of type  $D \rightarrow D \rightarrow \mathbb{P}$ , up to extensional equivalence. As all the iterated values  $F^n(\lambda uv. \top)$  are strongly decidable, they all belong to the weak powerset, so by Theorem 7, a duplicate is to be found in the first  $l + 1$  steps, ensuring that the sequence is stalled at  $l$ .  $\square$

We use the strongly decidable congruence  $\dot{=}$  to quotient models onto discrete ones (in fact  $\mathbb{F}_n$  for some  $n$ ) while preserving first-order satisfaction.

**Theorem 20.** *For every first-order signature  $\Sigma$  and formula  $\varphi$  over  $\Sigma$ , we have  $\text{FSAT}(\Sigma) \varphi$  iff  $\text{FSAT}'(\Sigma) \varphi$ , and as a consequence, both reductions  $\text{FSAT}(\Sigma) \preceq \text{FSAT}'(\Sigma)$  and  $\text{FSAT}'(\Sigma) \preceq \text{FSAT}(\Sigma)$  hold.*

*Proof.*  $\text{FSAT}(\Sigma) \varphi$  entails  $\text{FSAT}'(\Sigma) \varphi$  is the non-trivial implication. Hence we consider a finite  $\Sigma$ -model  $(D, \mathcal{M}, \rho)$  of  $\varphi$  and we build a new finite  $\Sigma$ -model of  $\varphi$  which is furthermore discrete. We collect the symbols occurring in  $\varphi$  as the lists  $l_{\mathcal{F}} := \mathcal{F}_{\varphi}$  (for functions) and  $l_{\mathcal{P}} := \mathcal{P}_{\varphi}$  (for relations). By Theorem 19, first-order indistinguishability  $\dot{=} : D \rightarrow D \rightarrow \mathbb{P}$  up to  $\mathcal{F}_{\varphi}/\mathcal{P}_{\varphi}$  is a strongly decidable equivalence over  $D$  and a congruence for the semantics of the symbols occurring in  $\varphi$ . Using Theorem 9, we build the quotient  $D/\dot{=}$  on a  $\mathbb{F}_n$  for some  $n : \mathbb{N}$ . We transport the model  $\mathcal{M}$  along this quotient and because  $\dot{=}$  is a congruence for the symbols in  $\varphi$ , its semantics is preserved along the quotient. Hence,  $\varphi$  has a finite model over the domain  $\mathbb{F}_n$  which is both finite and discrete.  $\square$

**Theorem 21.** *If  $\equiv$  is a binary relation symbol in the signature  $\Sigma$ , one has a reduction  $\text{FSATEQ}(\Sigma; \equiv) \preceq \text{FSAT}(\Sigma)$ .*

*Proof.* Given a list  $l_{\mathcal{F}}$  (resp.  $l_{\mathcal{P}}$ ) of function (resp. relation) symbols, we construct a formula  $\psi(l_{\mathcal{F}}, l_{\mathcal{P}}, \equiv)$  over the function symbols in  $l_{\mathcal{F}}$  and relation symbols in  $(\equiv :: l_{\mathcal{P}})$  expressing the requirement that  $\equiv$  is an equivalence and a congruence for the symbols in  $l_{\mathcal{F}}/l_{\mathcal{P}}$ . Then we show that  $\lambda\varphi. \varphi \dot{\wedge} \psi(\mathcal{F}_{\varphi}, \equiv :: \mathcal{P}_{\varphi}, \equiv)$  is a correct reduction, where  $\mathcal{F}_{\varphi}$  and  $\mathcal{P}_{\varphi}$  list the symbols occurring in  $\varphi$ .  $\square$

## 4.2 From Discrete Signatures to Singleton Signatures

Let us start by converting a discrete signature to a finite and discrete signature.

**Lemma 22.** *For any formula  $\varphi$  over a discrete signature  $\Sigma$ , one can compute a signature  $\Sigma_{n,m} = (\mathbb{F}_n; \mathbb{F}_m)$ , arity preserving maps  $\mathbb{F}_n \rightarrow \mathcal{F}_{\Sigma}$  and  $\mathbb{F}_m \rightarrow \mathcal{P}_{\Sigma}$  and an equi-satisfiable formula  $\psi$  over  $\Sigma_{n,m}$ , i.e.  $\text{FSAT}(\Sigma) \varphi \leftrightarrow \text{FSAT}(\Sigma_{n,m}) \psi$ .*

*Proof.* We use the discreteness of  $\Sigma$  and bijectively map the lists of symbols  $\mathcal{F}_{\varphi}$  and  $\mathcal{P}_{\varphi}$  onto  $\mathbb{F}_n$  and  $\mathbb{F}_m$  respectively, using Corollary 10. We structurally map  $\varphi$  to  $\psi$  over  $\Sigma_{n,m}$  along this bijection, which preserves finite satisfiability.  $\square$

Notice that  $n$  and  $m$  in the signature  $\Sigma_{n,m}$  depend on  $\varphi$ , hence the above statement cannot be presented as a reduction between (fixed) signatures.

We now erase all function symbols by encoding them with relation symbols. To this end, let  $\Sigma = (\mathcal{F}_{\Sigma}; \mathcal{P}_{\Sigma})$  be a signature, we set  $\Sigma' := (\emptyset; \{\equiv^2\} + \mathcal{F}_{\Sigma}^{+1} + \mathcal{P}_{\Sigma})$  where  $\equiv$  is a new interpreted relation symbol of arity two and in the conversion, function symbols have arity lifted by one, hence the  $\mathcal{F}_{\Sigma}^{+1}$  notation.

**Lemma 23.** *For any finite<sup>13</sup> type of function symbols  $\mathcal{F}_{\Sigma}$ , one has a reduction  $\text{FSAT}'(\mathcal{F}_{\Sigma}; \mathcal{P}_{\Sigma}) \preceq \text{FSATEQ}(\emptyset; \{\equiv^2\} + \mathcal{F}_{\Sigma}^{+1} + \mathcal{P}_{\Sigma}; \equiv^2)$ .*

*Proof.* The idea is to recursively replace a term  $t$  over  $\Sigma$  by a formula which is “equivalent” to  $x \equiv t$  (where  $x$  is a fresh variable not occurring in  $t$ ) and then an atomic formula like e.g.  $P[t_1; t_2]$  by  $\exists x_1 x_2. x_1 \equiv t_1 \dot{\wedge} x_2 \equiv t_2 \dot{\wedge} P[x_1; x_2]$ . We complete the encoding with a formula stating that every function symbol  $f : \mathcal{F}_{\Sigma}$  is encoded into a total functional relation  $P_f : \mathcal{F}_{\Sigma}^{+1}$  of arity augmented by 1.  $\square$

Next, assuming that the function symbols have already been erased, we explain how to merge the relation symbols in a signature  $\Sigma = (\emptyset; \mathcal{P}_{\Sigma})$  into a single relation symbol, provided that there is an upper bound for the arities in  $\mathcal{P}_{\Sigma}$ .

**Lemma 24.** *The reduction  $\text{FSAT}(\emptyset; \mathcal{P}_{\Sigma}) \preceq \text{FSAT}(\emptyset; \{Q^{1+n}\})$  holds when  $\mathcal{P}_{\Sigma}$  is a finite and discrete type of relation symbols and  $|P| \leq n$  holds for all  $P : \mathcal{P}_{\Sigma}$ .*

*Proof.* This comprises three independent reductions, see Fact 25 below.  $\square$

<sup>13</sup> In the Coq code, we prove the theorem for finite *or* discrete types of function symbols.

In the following, we denote by  $\mathcal{F}_\Sigma^n$  (resp.  $\mathcal{P}_\Sigma^n$ ) the same type of function (resp. relation) symbols but where the arity is uniformly converted to  $n$ .

**Fact 25.** *Let  $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$  be a signature:*

1.  $\text{FSAT}(\mathcal{F}_\Sigma; \mathcal{P}_\Sigma) \preceq \text{FSAT}(\mathcal{F}_\Sigma; \mathcal{P}_\Sigma^n)$  if  $|P| \leq n$  holds for all  $P : \mathcal{P}_\Sigma$ ;
2.  $\text{FSAT}(0; \mathcal{P}_\Sigma^n) \preceq \text{FSAT}(\mathcal{P}_\Sigma^0; \{Q^{1+n}\})$  if  $\mathcal{P}_\Sigma$  is finite;
3.  $\text{FSAT}(\mathcal{F}_\Sigma^0; \mathcal{P}_\Sigma) \preceq \text{FSAT}(0; \mathcal{P}_\Sigma)$  if  $\mathcal{F}_\Sigma$  is discrete.

*Proof.* For the first reduction, every atomic formula of the form  $P \mathbf{v}$  with  $|\mathbf{v}| = |P| \leq n$  is converted to  $P \mathbf{v}'$  with  $\mathbf{v}' := \mathbf{v} \# [x_0; \dots; x_0]$  and  $|\mathbf{v}'| = n$  for an arbitrary term variable  $x_0$ . The rest of the structure of formulas is unchanged.

For the second reduction, we convert every atomic formula  $P \mathbf{v}$  with  $|\mathbf{v}| = n$  into  $Q(P :: \mathbf{v})$  where  $P$  now represents a constant symbol ( $Q$  is fixed).

For the last reduction, we replace every constant symbol by a corresponding fresh variable chosen above all the free variables of the transformed formula.  $\square$

### 4.3 Compressing $n$ -ary Relations to Binary Membership

Let  $\Sigma_n = (0; \{P^n\})$  be a singleton signature where  $P$  is of arity  $n$ . We now show that  $P$  can be compressed to a binary relation modelling membership via a construction using hereditarily finite sets [18] (useful only when  $n \geq 3$ ).

**Theorem 26.**  $\text{FSAT}'(0; \{P^n\}) \preceq \text{FSAT}(0; \{\dot{\in}^2\})$ .

Technically, this reduction is one of the most involved in this work, although in most presentations of Trakhtenbrot’s theorem, this is left as an “easy exercise”; see e.g. [14]. Maybe it is perceived so because it relies on the encoding of tuples in set theory, which is somehow natural for mathematicians,<sup>14</sup> but properly building the finite set model in constructive type theory was not that easy.

Here we only give an overview of the main tools. We encode an arbitrary  $n$ -ary relation  $R : X^n \rightarrow \mathbb{P}$  over a finite type  $X$  in the theory of *membership* over the signature  $\Sigma_2 = (0; \{\dot{\in}^2\})$ . Membership is much weaker than set theory because the only required set-theoretic axiom is *extensionality*. Two sets are extensionally equal if their members are the same, and extensionality states that two extensionally equal sets belong to the same sets:

$$\dot{\forall}xy. (\dot{\forall}z. z \dot{\in} x \leftrightarrow z \dot{\in} y) \dot{\rightarrow} \dot{\forall}z. x \dot{\in} z \dot{\rightarrow} y \dot{\in} z \quad (1)$$

As a consequence, no first-order formula over  $\Sigma_2$  can distinguish two extensionally equal sets. Notice that the language of membership theory (and set theory) does not contain any function symbol, hence, contrary to usual mathematical practices, there is no other way to handle a set than via its characterising formula which makes it a very cumbersome language to work with formally. However, this is how we have to proceed in the Coq development but here, we stick to meta-level “terms” in the prose for simplicity.

<sup>14</sup> In our case we use Kuratowski’s encoding.

The ordered pair of two sets  $p$  and  $q$  is encoded as  $(p, q) := \{\{p\}, \{p, q\}\}$  while the  $n$ -tuple  $(t_1, \dots, t_n)$  is encoded as  $(t_1, (t_2, \dots, t_n))$  recursively. The reduction function which maps formulas over  $\Sigma_n$  to formulas over  $\Sigma_2$  proceeds as follows. We reserve two first-order variables  $d$  (for the domain  $D$ ) and  $r$  (for the relation  $R$ ). We describe the recursive part of the reduction  $\Sigma_{n \rightsquigarrow 2}^r$

$$\begin{aligned} \Sigma_{n \rightsquigarrow 2}^r(P \mathbf{v}) &:= \text{“tuple } \mathbf{v} \in r\text{”} & \Sigma_{n \rightsquigarrow 2}^r(\dot{\forall}z. \varphi) &:= \dot{\forall}z. z \in d \dot{\rightarrow} \Sigma_{n \rightsquigarrow 2}^r(\varphi) \\ \Sigma_{n \rightsquigarrow 2}^r(\varphi \dot{\square} \psi) &:= \Sigma_{n \rightsquigarrow 2}^r(\varphi) \dot{\square} \Sigma_{n \rightsquigarrow 2}^r(\psi) & \Sigma_{n \rightsquigarrow 2}^r(\dot{\exists}z. \varphi) &:= \dot{\exists}z. z \in d \dot{\wedge} \Sigma_{n \rightsquigarrow 2}^r(\varphi) \end{aligned}$$

ignoring the de Bruijn syntax (which would imply adding  $d$  and  $r$  as parameters). Notice that  $d$  and  $r$  should not occur freely in  $\varphi$ . In addition, we require that:

$$\begin{aligned} \varphi_1 &:= \dot{\in} \text{ is extensional} && \text{see Equation (1);} \\ \varphi_2 &:= \dot{\exists}z. z \in d && \text{i.e. } d \text{ is non-empty;} \\ \varphi_3 &:= x_1 \in d \dot{\wedge} \dots \dot{\wedge} x_k \in d && \text{where } [x_1; \dots; x_k] = \text{FV}(\varphi). \end{aligned}$$

This gives us the reduction function  $\Sigma_{n \rightsquigarrow 2}(\varphi) := \varphi_1 \dot{\wedge} \varphi_2 \dot{\wedge} \varphi_3 \dot{\wedge} \Sigma_{n \rightsquigarrow 2}^r(\varphi)$ .

The *completeness* of the reduction  $\Sigma_{n \rightsquigarrow 2}$  is the easy part. Given a finite model of  $\Sigma_{n \rightsquigarrow 2}(\varphi)$  over  $\Sigma_2$ , we recover a model of  $\varphi$  over  $\Sigma_n$  by selecting as the new domain the members of  $d$  and the interpretation of  $P \mathbf{v}$  is given by testing whether the encoding of  $\mathbf{v}$  as a  $n$ -tuple is a member of  $r$ .

The *soundness* of the reduction  $\Sigma_{n \rightsquigarrow 2}$  is the formally involved part, with Theorem 27 below containing the key construction.

**Theorem 27.** *Given a decidable  $n$ -ary relation  $R : X^n \rightarrow \mathbb{P}$  over a finite, discrete and inhabited type  $X$ , one can compute a finite and discrete type  $Y$  equipped with a decidable relation  $\in : Y \rightarrow Y \rightarrow \mathbb{P}$ , two distinguished elements  $d, r : Y$  and a pair of maps  $i : X \rightarrow Y$  and  $s : Y \rightarrow X$  s.t.*

1.  $\in$  is extensional;
2. extensionally equal elements of  $Y$  are equal;
3. all  $n$ -tuples of members of  $d$  exist in  $Y$ ;
4.  $\forall x : X. i x \in d$ ;
5.  $\forall y : Y. y \in d \rightarrow \exists x. y = i x$ ;
6.  $\forall x : X. s(i x) = x$ ;
7.  $R \mathbf{v}$  iff  $i(\mathbf{v})$  is a  $n$ -tuple member of  $r$ , for any  $\mathbf{v} : X^n$ .

*Proof.* We give a brief outline of this quite involved proof, referring to the Coq code for details. The type  $Y$  is built from the type of hereditarily finite sets based on [18], and when we use the word “set” below, it means hereditarily finite set. The idea is first to construct  $d$  as a *transitive set* of which the elements are in bijection  $i/s$  with the type  $X$ , hence  $d$  is the cardinal of  $X$  in the set-theoretic meaning. Then the iterated powersets  $\mathcal{P}(d), \mathcal{P}^2(d), \dots, \mathcal{P}^k(d)$  are all transitive as well and contain  $d$  both as a member and as a subset. Considering  $\mathcal{P}^{2n}(d)$  which contains all the  $n$ -tuples built from the members of  $d$ , we define  $r$  as the set of  $n$ -tuples collecting the encodings  $i(\mathbf{v})$  of vectors  $\mathbf{v} : X^n$  such that  $R \mathbf{v}$ . We show  $r \in p$  for  $p$  defined as  $p := \mathcal{P}^{2n+1}(d)$ . Using the Boolean counterpart of  $(\cdot) \in p$  for unicity of proofs, we then define  $Y := \{z \mid z \in p\}$ , restrict membership  $\in$  to  $Y$  and this gives the finite type equipped with all the required properties. Notice that the decidability requirement for  $\in$  holds constructively because we work with hereditarily finite sets, and would not hold with arbitrary sets.  $\square$

#### 4.4 Summary: From Discrete Signatures to the Binary Signature

Combining all the previous results, we give a reduction from any discrete signature to the binary singleton signature.

**Theorem 28.**  $\text{FSAT}(\Sigma) \preceq \text{FSAT}(0; \{P^2\})$  holds for any discrete signature  $\Sigma$ .

*Proof.* Let us first consider the case of  $\Sigma_{n,m} = (\mathbb{F}_n; \mathbb{F}_m)$ , a signature over the finite and discrete types  $\mathbb{F}_n$  and  $\mathbb{F}_m$ . Then we have a reduction  $\text{FSAT}(\mathbb{F}_n; \mathbb{F}_m) \preceq \text{FSAT}(0; \{P^2\})$  by combining Theorems 20, 21 and 26 and Lemmas 23 and 24.

Let us denote by  $f_{n,m}$  the reduction  $\text{FSAT}(\mathbb{F}_n; \mathbb{F}_m) \preceq \text{FSAT}(0; \{P^2\})$ . Let us now consider a fixed discrete signature  $\Sigma$ . For a formula  $\varphi$  over  $\Sigma$ , using Lemma 22, we compute a signature  $\Sigma_{n,m}$  and  $\psi$  over  $\Sigma_{n,m}$  s.t.  $\text{FSAT}(\Sigma) \varphi \leftrightarrow \text{FSAT}(\mathbb{F}_n; \mathbb{F}_m) \psi$ . The map  $\lambda\varphi.f_{n,m} \psi$  is the required reduction.  $\square$

**Lemma 29.**  $\text{FSAT}(0; \{P^2\}) \preceq \text{FSAT}(\{f^n\}; \{Q^1\})$  when  $n \geq 2$ .

*Proof.* We encode the binary relation  $\lambda x y. P[x; y]$  with  $\lambda x y. Q(f[x; y; \dots])$ , using the first two parameters of  $f$  to encode pairing. But since we need to change the domain of the model, we also use a fresh variable  $d$  to encode the domain as  $\lambda x. Q(f[d; x; \dots])$  and we restrict all quantifications to the domain similarly to the encoding  $\Sigma_{n \rightsquigarrow 2}^r$  of Section 4.3.  $\square$

We finish the reduction chains with the weakest possible signature constraints. The following reductions have straightforward proofs.

**Fact 30.** One has reductions for the three statements below (for  $n \geq 2$ ):

1.  $\text{FSAT}(0; \{P^2\}) \preceq \text{FSAT}(0; \{P^n\})$ ;
2.  $\text{FSAT}(0; \{P^n\}) \preceq \text{FSAT}(\Sigma)$  if  $\Sigma$  contains an  $n$ -ary relation symbol;
3.  $\text{FSAT}(\{f^n\}; \{Q^1\}) \preceq \text{FSAT}(\Sigma)$  if  $\Sigma$  contains an  $n$ -ary fun. and a unary rel.

## 5 Decidability Results

Complementing the previously studied negative results, we now examine the conditions allowing for decidable satisfiability problems.

**Lemma 31. (FSAT over a fixed domain)** Given a discrete signature  $\Sigma$  and a discrete and finite type  $D$ , one can decide whether or not a formula over  $\Sigma$  has a (finite) model over domain  $D$ .

*Proof.* By Fact 12, satisfaction in a given finite model is decidable. It is also invariant under extensional equivalence, so we only need to show that there are finitely many (decidable) models over  $D$  up to extensional equivalence!<sup>15</sup>  $\square$

**Lemma 32.** A formula over a signature  $\Sigma$  has a finite and discrete model if and only if it has a (finite) model over  $\mathbb{F}_n$  for some  $n : \mathbb{N}$ .

<sup>15</sup> Without discreteness of  $\Sigma$ , it is impossible to build the list of models over  $D = \mathbb{B}$ .

*Proof.* If  $\varphi$  has a model over a discrete and finite domain  $D$ , by Corollary 10, one can bijectively map  $D$  to  $\mathbb{F}_n$  and transport the model along this bijection.  $\square$

**Lemma 33.** *FSAT( $(0; \mathcal{P}_\Sigma)$ ) is decidable if  $\mathcal{P}_\Sigma$  is discrete with uniform arity 1.*

*Proof.* By Lemma 22, we can assume  $\mathcal{P}_\Sigma = \mathbb{F}_n$  w.l.o.g. We show that if  $\varphi$  has a finite model then it must have a model over domain  $\{\mathbf{v} : \mathbb{B}^n \rightarrow \mathbb{B} \mid b\mathbf{v} = \mathbf{tt}\}$  for some Boolean subset  $b : (\mathbb{B}^n \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$ . Up to extensional equivalence, there are only finitely many such subsets  $b$  and we conclude with Lemma 31.  $\square$

**Lemma 34.** *For any finite type  $\mathcal{P}_\Sigma$  of relation symbols and signatures of uniform arity 1, we have a reduction  $\text{FSAT}(\mathbb{F}_n; \mathcal{P}_\Sigma) \preceq \text{FSAT}(0; \mathbb{L}\mathbb{F}_n \times \mathcal{P}_\Sigma + \mathcal{P}_\Sigma)$ .*

*Proof.* We implemented a proof somewhat inspired by that of Proposition 6.2.7 (Grädel) in [1, pp. 251] but the invariant suggested in the iterative process described there did not work out formally and we had to proceed in a single conversion step instead, switching from single symbols to lists of symbols.  $\square$

If functions or relations have arity 0, one can always lift them to arity 1 using a fresh variable (of arbitrary value), like in Fact 25, item (1).

**Fact 35.** *The reduction  $\text{FSAT}(\mathcal{F}_\Sigma; \mathcal{P}_\Sigma) \preceq \text{FSAT}(\mathcal{F}_\Sigma^1; \mathcal{P}_\Sigma^1)$  holds when all arities in  $\Sigma$  are at most 1, where  $\mathcal{F}_\Sigma^1$  and  $\mathcal{P}_\Sigma^1$  denote arities uniformly updated to 1.*

## 6 Signature Classification

We conclude with the exact classification of FSAT regarding enumerability, decidability, and undecidability depending on the properties of the signature.

**Theorem 36.** *Given  $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma)$  where both  $\mathcal{F}_\Sigma$  and  $\mathcal{P}_\Sigma$  are data types, the finite satisfiability problem for formulas over  $\Sigma$  is enumerable.*

*Proof.* Using Theorem 20 and Lemmas 31 and 32, one constructs a predicate  $Q : \mathbb{N} \rightarrow \text{Form}_\Sigma \rightarrow \mathbb{B}$  s.t.  $\text{FSAT}(\Sigma) \varphi \leftrightarrow \exists n. Q n \varphi = \mathbf{tt}$ . Then, it is easy to build a computable enumeration  $e : \mathbb{N} \rightarrow \odot \text{Form}_\Sigma$  of  $\text{FSAT}(\Sigma) : \text{Form}_\Sigma \rightarrow \mathbb{P}$ .  $\square$

**Theorem 37. (Full Monadic FOL)** *FSAT( $\Sigma$ ) is decidable if  $\Sigma$  is discrete with arities less or equal than 1, or if all relation symbols have arity 0.*

*Proof.* If all arities are at most 1, then by Fact 35, we can assume  $\Sigma$  of uniform arity 1. Therefore, for a formula  $\varphi$  over  $\Sigma$  with uniform arity 1, we need to decide FSAT for  $\varphi$ . By Theorem 22, we can compute a signature  $\Sigma_{n,m} = (\mathbb{F}_n; \mathbb{F}_m)$  and a formula  $\psi$  over  $\Sigma_{n,m}$  equi-satisfiable with  $\varphi$ . Using the reduction of Lemma 34, we compute a formula  $\gamma$ , equi-satisfiable with  $\psi$ , over a discrete signature of uniform arity 1, void of functions. We decide the satisfiability of  $\gamma$  by Lemma 33.

If all relation symbols have arity 0, regardless of  $\mathcal{F}_\Sigma$ , no term can occur in formulas, hence neither can function symbols. Starting from  $\varphi$  over  $\Sigma = (\mathcal{F}_\Sigma; \mathcal{P}_\Sigma^0)$  where only  $\mathcal{P}_\Sigma$  is assumed discrete, we compute an equi-satisfiable formula  $\psi$  over  $\Sigma' = (0; \mathcal{P}_\Sigma^0)$  and we are back to the previous case.  $\square$

**Theorem 38. (Full Trakhtenbrot)** *If  $\Sigma$  contains either an at least binary relation symbol or a unary relation symbol together with an at least binary function symbol, then BPCP reduces to FSAT( $\Sigma$ ).*

*Proof.* By Theorems 14, 21 and 28, Lemma 29, and Fact 30. □

**Corollary 39.** *For an enumerable and discrete signature  $\Sigma$  furthermore satisfying the conditions in Theorem 38, FSAT( $\Sigma$ ) is both enumerable and undecidable, thus, more specifically, not co-enumerable.*

*Proof.* Follows by Facts 3 and 5. □

Notice that even if the conditions on arities of Theorems 37 and 38 fully classify discrete signatures, it is not possible to decide which case holds unless the signature is furthermore finite. For a given formula  $\varphi$  though, it is always possible to render it in the finite signature of used symbols.

## 7 Discussion

The main part of our Coq development directly concerned with the classification of finite satisfiability consists of 10k loc, in addition to 3k loc of (partly reused) utility libraries. Most of the code comprises the signature transformations with more than 4k loc for reducing discrete signatures to membership. Comparatively, the initial reduction from BPCP to FSATEQ( $\Sigma_{\text{BPCP}}$ ) takes less than 500 loc.

Our mechanisation of first-order logic in principle follows previous developments [8,9] but also differs in a few aspects. Notably, we had to separate function from relation signatures to be able to express distinct signatures that agree on one sort of symbols computationally. Moreover, we found it favourable to abstract over the logical connectives in form of  $\Box$  and  $\nabla$  to shorten purely structural definitions and proofs. Finally, we did not use the Autosubst 2 [20] support for de Bruijn syntax to avoid its current dependency on the functional extensionality axiom.

We refrained from additional axioms since we included our development in the growing Coq library of synthetic undecidability proofs [11]. In this context, we plan to generalise some of the intermediate signature reductions so that they become reusable for other undecidability proofs concerning first-order logic over arbitrary models.

As further future directions, we want to explore and mechanise the direct consequences of Trakhtenbrot's theorem such as the undecidability of query containment and equivalence in data base theory or the undecidability of separation logic [3,5]. Also possible, though rather ambitious, would be to mechanise the classification of first-order satisfiability with regards to the quantifier prefix as comprehensively developed in [1]. Finally, we plan to mechanise the undecidability of semantic entailment and syntactic deduction in first-order axiom systems such as ZF set theory and Peano arithmetic.



## References

1. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic, Springer-Verlag Berlin Heidelberg (1997)
2. Braibant, T., Pous, D.: An efficient Coq tactic for deciding Kleene algebras. In: International Conference on Interactive Theorem Proving. pp. 163–178. Springer (2010)
3. Brochenin, R., Demri, S., Lozes, E.: On the almighty wand. *Information and Computation* **211**, 106–137 (2012)
4. de Bruijn, N.G.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)* **75**(5), 381–392 (Jan 1972)
5. Calcagno, C., Yang, H., O’Hearn, P.W.: Computability and Complexity Results for a Spatial Assertion Language for Data Structures. In: Hariharan, R., Vinay, V., Mukund, M. (eds.) *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science*. pp. 108–119. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
6. Cohen, C.: Pragmatic Quotient Types in Coq. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) *Interactive Theorem Proving*. pp. 213–228. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
7. Forster, Y., Heiter, E., Smolka, G.: Verification of PCP-related computational reductions in Coq. In: *International Conference on Interactive Theorem Proving*. pp. 253–269. Springer (2018)
8. Forster, Y., Kirst, D., Smolka, G.: On synthetic undecidability in Coq, with an application to the Entscheidungsproblem. In: *International Conference on Certified Programs and Proofs*. pp. 38–51. ACM (2019)
9. Forster, Y., Kirst, D., Wehr, D.: Completeness Theorems for First-Order Logic Analysed in Constructive Type Theory. In: *Symposium on Logical Foundations Of Computer Science, 2020, Deerfield Beach, Florida, U.S.A.* (Jan 2020)
10. Forster, Y., Larchey-Wendling, D.: Certified Undecidability of Intuitionistic Linear Logic via Binary Stack Machines and Minsky Machines. In: *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*. pp. 104–117. ACM (2019)
11. Forster, Y., Larchey-Wendling, D., Dudenhefner, A., Heiter, E., Kirst, D., Kunze, F., Smolka, G., Spies, S., Wehr, D., Wuttke, M.: A Coq Library of Undecidable Problems. In: *CoqPL 2020, New Orleans, LA, United States (2020)*, <https://github.com/uds-psl/coq-library-undecidability>
12. Kirst, D., Larchey-Wendling, D.: Trakhtenbrot’s Theorem in Coq, A Constructive Approach to Finite Model Theory (2020), <https://arxiv.org/abs/2004.07390>
13. Larchey-Wendling, D., Forster, Y.: Hilbert’s Tenth Problem in Coq. In: *4th International Conference on Formal Structures for Computation and Deduction. LIPIcs*, vol. 131, pp. 27:1–27:20 (Feb 2019)
14. Libkin, L.: *Elements of Finite Model Theory*. Springer Publishing Company, Incorporated, 1st edn. (2010)
15. Löwenheim, L.: Über Möglichkeiten im Relativkalkül. *Mathematische Annalen* **76**, 447–470 (1915), <http://eudml.org/doc/158703>
16. Maksimović, P., Schmitt, A.: HOCore in Coq. In: *International Conference on Interactive Theorem Proving*. pp. 278–293. Springer (2015)
17. Schäfer, S., Smolka, G., Tebbi, T.: Completeness and decidability of de Bruijn substitution algebra in Coq. In: *Proceedings of the 2015 Conference on Certified Programs and Proofs*. pp. 67–73. ACM (2015)

18. Smolka, G., Stark, K.: Hereditarily Finite Sets in Constructive Type Theory. In: Interactive Theorem Proving - 7th International Conference, ITP 2016, Nancy, France, August 22-27, 2016. LNCS, vol. 9807, pp. 374–390. Springer (2016)
19. Spies, S., Forster, Y.: Undecidability of Higher-Order Unification Formalised in Coq. In: International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, USA (Jan 2020)
20. Stark, K., Schäfer, S., Kaiser, J.: Autosubst 2: reasoning with multi-sorted de Bruijn terms and vector substitutions. In: International Conference on Certified Programs and Proofs. pp. 166–180. ACM (2019)
21. Trakhtenbrot, B.A.: The impossibility of an algorithm for the decidability problem on finite classes. Dokl. Akad. Nauk. SSSR **70**(4), 569–572 (1950)
22. Veldman, W., Waaldijk, F.: Some Elementary Results in Intuitionistic Model Theory. The Journal of Symbolic Logic **61**(3), 745–767 (1996)
23. Werner, B.: Sets in Types, Types in Sets. In: Theoretical Aspects of Computer Software. pp. 530–546. Springer, Berlin, Heidelberg (Sep 1997)