Hilbert's Tenth Problem in Coq

Dominique Larchey-Wendling
Université de Lorraine, CNRS, LORIA, Vandœuvre-lès-Nancy, France
dominique.larchey-wendling@loria.fr

Yannick Forster
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
forster@ps.uni-saarland.de

Abstract

We formalise the undecidability of solvability of Diophantine equations, i.e. polynomial equations over natural numbers, in Coq's constructive type theory. To do so, we give the first full mechanisation of the Davis-Putnam-Robinson-Matiyasevich theorem, stating that every recursively enumerable problem – in our case by a Minsky machine – is Diophantine. We obtain an elegant and comprehensible proof by using a synthetic approach to computability and by introducing Conway's FRACTRAN language as intermediate layer.

2012 ACM Subject Classification Theory of computation → Models of computation; Type theory

Keywords and phrases Hilbert's tenth problem, diophantine equations, undecidability, computability theory, reduction, Coq, type theory

Supplement Material Coq formalisation of all results: https://uds-psl.github.io/H10
Coq library of undecidable problems: https://github.com/uds-psl/coq-library-undecidability

1 Introduction

Hilbert's tenth problem (H10) was posed by David Hilbert in 1900 as part of his famous 23 problems [14] and asked for the "determination of the solvability of a Diophantine equation." A Diophantine equation is a polynomial equation over natural numbers (or, equivalently, integers) with constant exponents, e.g. $x^2+y+3z = 0$. When Hilbert asked for "determination," he meant, in modern terms, a decision procedure, but computability theory was yet several decades short of being developed.

The first undecidable problems found by Church, Post and Turing were either native to mathematical logic or dependent on a fixed model of computation. H10, to the contrary, can be stated to every mathematician and its formulation is independent from a model of computation. Emil Post stated in 1944 that H10 "begs for an unsolvability proof" [25]. From a computational perspective, it is clear that H10 is recursively enumerable (or *recognisable*), meaning there is an algorithm that halts on a Diophantine equation iff it is solvable.

Post's student Martin Davis conjectured that even the converse is true, i.e. that every recognisable set is also Diophantine. More precisely, he conjectured that if $A \subseteq \mathbb{N}^k$ is recognisable then $(a_1, \dots, a_k) \in A \leftrightarrow \exists x_1 \dots x_n, P(a_1, \dots, a_k, x_1, \dots, x_n) = 0$ holds for some polynomial P in $k + n$ variables. He soon improved on a result by Gödel [12] and gave a proof of his conjecture up to one bounded universal quantification [3]: $(a_1, \dots, a_k) \in A \leftrightarrow \exists z, \forall y < z, \exists x_1 \dots x_n, P(a_1, \dots, a_k, x_1, \dots, x_n, y, z) = 0$. Davis and Putnam [4] further improved on this, and showed that, provided a certain number-theoretic assumption holds, every recognisable set is *exponentially* Diophantine, meaning variables are also allowed to appear in exponents. Julia Robinson then in 1961 modified the original proof to circumvent the need for the assumption, resulting in the DPR theorem [5], namely that every recognisable set is exponentially Diophantine. Due to another result from Robinson [26], the gap now only consisted of proving that there is a Diophantine equation exhibiting exponential growth. In 1970, Yuri Matiyasevich showed that the Fibonacci sequence grows exponentially while

being Diophantine, closing the gap and finishing the proof of the theorem nowadays called *DPRM theorem*, ultimately establishing that exponentiation is Diophantine itself [18] (known as "Matiyasevich's theorem").

Even the most modern and simpler proofs of the DPRM theorem still require many preliminaries and complicated number-theoretic ideas, for an overview see [21]. We formalise one such proof as part of our ongoing work on a library of undecidable problems [9] in the proof assistant Coq [28]. Since H10 is widely used as a seed for showing the undecidability of problems using *many-one reductions* [6, 13], this will open further ways of extending the library. Given that our library already contains a formalisation of Minsky machines [10], we follow the approach of Jones and Matijasevič [15], who use register machines, being very well-suited since they already work on numbers. They encode full computations of register machines as Diophantine equations in one single, monolithic step. To make the proof more tractable for both implementation and explanation, we factor out an intermediate language, John Conway's FRACTRAN [2], which can simulate Minsky machines.

We first introduce three characterisations of Diophantine equations over natural numbers, namely *Diophantine logic* DIO_FORM (allowing to connect Diophantine equations with conjunction, disjunction and existential quantification), *elementary Diophantine constraints* DIO_ELEM (a finite set of constraints on variables, oftentimes used for reductions [6, 13]) and *single Diophantine equations* DIO_SINGLE, including parameters, as described above. H10 then asks about the solvability of single Diophantine equations with no parameters.

Technically, the reduction chain to establish the unsolvability of H10 starts at the halting problem for single-tape Turing machines Halt, reduced to the Post correspondence problem PCP in [7]. In previous work [10] we have reduced PCP to a specialised halting problem for Minsky machines, which we use here in a slighly generalised form as MM. We then reduce Minsky machine halting to FRACTRAN termination. FRACTRAN is very natural to describe using polynomials, and the encoding does not rely on any complicated construction. The technical difficulty then only lies in the Diophantine encoding of the reflexive-transitive closure of a relation which follows from the direct elimination of bounded universal quantification, given that the proof in [19] involves no detour via models of computation. In total, we obtain the following chain of reductions to establish the undecidability of H10:

$$\mathsf{Halt} \preceq \mathsf{PCP} \preceq \mathsf{MM} \preceq \mathsf{FRACTRAN} \preceq \mathsf{DIO\_FORM} \preceq \mathsf{DIO\_ELEM} \preceq \mathsf{DIO\_SINGLE} \preceq \mathsf{H10}$$

In the present paper, we focus on explaining this factorisation of the proof and give some details for the different stages. While we contribute Coq mechanisations of Matiyasevich's theorem and the elimination of bounded universal quantification, we treat them mainly as black-boxes and only elaborate on their challenging formalisation rather than the proofs themselves, a good explanation of which would anyways not fit in the given page limit.

To the best of our knowledge, we are the first to give a *full verification* of the DPRM theorem and the undecidability of Hilbert's tenth problem in a proof assistant. We base the notion of recognisability in the DPRM theorem on Minsky machines.

When giving undecidability proofs via many-one reductions, it is critical to show that all reduction functions are actually computable. We could in theory verify the computability of all functions involved using an explicit model of computation. In pen-and-paper proofs, this approach is however almost never used, because implementing high-level mathematical transformations as provably correct low-level programs is a daunting task. Instead, we rely on a synthetic approach [7, 8, 10] based on the computability of all functions definable in Coq's constructive type theory, which is closer to the practice of pen-and-paper proofs. In this approach, a problem P is considered undecidable if there is a reduction from an obviously undecidable problem, e.g. $\mathsf{Halt} \preceq P$.

The axiom-free Coq formalisation of all the results in this paper is available online and the main lemmas and theorems in the pdf version of the paper are hyper-linked with the html version of the source code at https://uds-psl.github.io/H10. Starting from our already existing library which included most of the Minsky machine code [10], the additional code for proving H10 and the DPRM consists in about 8k loc including 3k loc for Matiyasevich's results alone, together with a 4k loc addition to our shared libraries; see Appendix A for more details. The paper itself can be read without in-depth knowledge of Coq or type theory.

Contribution: Apart from the full formalisation, we consider the novel refactoring of the proof via FRACTRAN a contribution to the explainability of the DPRM theorem.

Preliminaries: Regarding notation, we may write $x.y$ for multiplication of natural numbers $x, y : \mathbb{N}$ and we will leave out the symbol where convenient. We write $\mathbb{L}\,X$ for the type of *lists* over X and $l +\!\!+ l'$ for the *concatenation* of two lists. We write X^n for *vectors* $\vec{v}$ over type X with length n, and $\mathbb{F}_n$ for the *finite type* with exactly n elements. For $p : \mathbb{F}_n$, we write $\vec{v}[p]$ for the p-th component of $\vec{v} : X^n$. Notations for lists are overloaded for vectors. If $P : X \to \mathbb{P}$ is a predicate (on X) and $Q : Y \to \mathbb{P}$ is a predicate, we write $P \preceq Q$ if there is a function $f : X \to Y$ s.t. $\forall x : X,\ P\,x \leftrightarrow Q(f\,x)$, i.e. a *many-one reduction* from P to Q.

2 Diophantine Relations

Diophantine relations are composed of polynomials over natural numbers. There are several equivalent approaches to characterise these relations and oftentimes, the precise definition is omitted from papers. Basically, one can form equations between polynomial expressions and then combine these with conjunctions, disjunctions, and existential quantification.[1] For instance, these operations are assumed as Diophantine producing operators in e.g. [15, 18, 19, 20]. Sometimes, Diophantine relations are restricted to a single polynomial equation. Sometimes, the exponentiation function $x, y \mapsto x^y$ is assumed as Diophantine [15]. To complicate the picture, Diophantine relations might equivalently range over $\mathbb{Z}$ (instead of $\mathbb{N}$) but expressions like x^y implicitly assume that y never gets a negative value.

Although seemingly diverging, these approaches are not contradictory because in the end, they characterise the same class of relations on natural numbers. However, mechanisation does not allow for such implicit assumptions. To give some mechanisable structure to some of these approaches, we propose three increasingly restricted characterisations of Diophantine relations: *Diophantine logic*, *elementary Diophantine constraints* and *single Diophantine equations*, between which we provide computable transformations in Sections 3 and 4.

2.1 Diophantine Logic

We define the types $\mathbb{D}_{\text{expr}}$ of Diophantine expressions and $\mathbb{D}_{\text{form}}$ of Diophantine formulæ for the abstract syntax of Diophantine logic. Diophantine expressions are *polynomials* built from natural number constants and variables. An atomic Diophantine logic formula is just expressing the identity between two Diophantine expressions and we combine those with binary disjunction, binary conjunction, and existential quantification.

$$p, q : \mathbb{D}_{\text{expr}} ::= x_i : \mathsf{V} \mid n : \mathbb{N} \mid p \mathbin{\dot{+}} q \mid p \mathbin{\dot{\times}} q \qquad A, B : \mathbb{D}_{\text{form}} ::= p \doteq q \mid A \mathbin{\dot{\wedge}} B \mid A \mathbin{\dot{\vee}} B \mid \dot{\exists}\, A$$

The letters p, q ranges over expressions and the letters A, B range over formulæ. We use standard *De Bruijn syntax* with variables $x_0, x_1, \ldots$ of type $\mathsf{V} := \mathbb{N}$ for better readability.

[1] Universal quantification or negation are not accepted as is.

If we have $x_i : \mathsf{V}$, we write x_{1+i} for the next variable in V. As an example, the meta-level formula $\exists y,\, y = 0 \wedge \exists z,\, y = z + 1$ would be represented as $\dot\exists(x_0 \doteq 1 \mathbin{\dot\wedge} \dot\exists(x_1 \doteq x_0 + 1))$, i.e. the variable x_i refers to the i-th binder in the context.

We provide a semantics for Diophantine logic. Given a valuation for variables $\nu : \mathsf{V} \to \mathbb{N}$, we define the interpretation $[\![p]\!]_\nu : \mathbb{N}$ of the expression $p : \mathbb{D}_{\text{expr}}$ by recursion:

$$[\![x_i]\!]_\nu := \nu\, x_i \qquad [\![n]\!]_\nu := n \qquad [\![p \mathbin{\dot+} q]\!]_\nu := [\![p]\!]_\nu + [\![q]\!]_\nu \qquad [\![p \mathbin{\dot\times} q]\!]_\nu := [\![p]\!]_\nu \times [\![q]\!]_\nu$$

The interpretation of formulæ cannot be done with a constant valuation $\nu : \mathsf{V} \to \mathbb{N}$ because of existential quantifiers. The interpretation $[\![A]\!]_\nu$ of the formula $A : \mathbb{D}_{\text{form}}$ is given by the following recursive rules:

$$\begin{array}{ll} [\![A \mathbin{\dot\wedge} B]\!]_\nu := [\![A]\!]_\nu \wedge [\![B]\!]_\nu & [\![p \doteq q]\!]_\nu := [\![p]\!]_\nu = [\![q]\!]_\nu \\ [\![A \mathbin{\dot\vee} B]\!]_\nu := [\![A]\!]_\nu \vee [\![B]\!]_\nu & [\![\dot\exists\, A]\!]_\nu := \exists n : \mathbb{N}, [\![A]\!]_{n\cdot\nu} \end{array} \qquad \text{with } \begin{cases} n{\cdot}\nu\,(x_0) := n \\ n{\cdot}\nu\,(x_{1+i}) := \nu\, x_i \end{cases}$$

where $n{\cdot}\nu : \mathsf{V} \to \mathbb{N}$ is the standard De Bruijn extension of a valuation ν by n.

We give a first formal characterisation of Diophantine polynomial expressions $\mathbb{D}_{\text{P}}$ and Diophantine relations $\mathbb{D}_{\text{R}}$. Diophantine polynomials are represented by some members of type $f : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ mapping valuations ν to values $f_\nu : \mathbb{N}$ which must moreover arise as instances of $\lambda\nu.[\![p]\!]_\nu$ for some $p : \mathbb{D}_{\text{expr}}$. And Diophantine relations are members of type $R : (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ arising as instances of $\lambda\nu.[\![A]\!]_\nu$. We give an informative content to these sub-types of $(\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ and $(\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ to be able to do some computations with the witness (either p or A) of Diophantineness, typically when moving to another formal representation like elementary Diophantine constraints in Section 3.

▶ **Definition 1**. *We define the class of* Diophantine polynomials *and* Diophantine relations *as informative sub-types of* $(\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ *and* $(\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ *respectively:*

$$\mathbb{D}_{\text{P}}\, f := \sum p : \mathbb{D}_{\text{expr}}, \big(\forall \nu, [\![p]\!]_\nu = f_\nu\big) \qquad \mathbb{D}_{\text{R}}\, R := \sum A : \mathbb{D}_{\text{form}}, \big(\forall \nu, [\![A]\!]_\nu \leftrightarrow R\ \nu\big)$$

Note that Σ denotes type-theoretic dependent pairs. Hence an inhabitant w of $\mathbb{D}_{\text{R}}\, R$ is a (dependent) pair (A, H_A) where $A = \pi_1(w)$ is a Diophantine formula and $H_A = \pi_2(w)$ a proof that $[\![A]\!]_{(\cdot)}$ and R are extensionally equivalent. With these definitions, we will show that the sub-types $\mathbb{D}_{\text{P}}$ and $\mathbb{D}_{\text{R}}$ have the desired closure properties: $\mathbb{D}_{\text{P}}$ contains variables, constants and is closed under the $+$ and $\times$ pointwise operators over $(\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$; $\mathbb{D}_{\text{R}}$ contains polynomial equations and is closed under conjunction, disjunction and existential quantification.

▶ **Proposition 2**. *Let* $x_i : \mathsf{V}$, $n : \mathbb{N}$, *and* $f, g : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ *be s.t.* $\mathbb{D}_{\text{P}}\, f$ *and* $\mathbb{D}_{\text{P}}\, g$ *hold. Then* $\mathbb{D}_{\text{P}}\, (\lambda\nu.\nu\, x_i)$, $\mathbb{D}_{\text{P}}\, (\lambda\nu.n)$, $\mathbb{D}_{\text{P}}\, (\lambda\nu.f_\nu + g_\nu)$ *and* $\mathbb{D}_{\text{P}}\, (\lambda\nu.f_\nu \times g_\nu)$ *hold.*

▶ **Proposition 3**. *Let* f, g *be s.t.* $\mathbb{D}_{\text{P}}\, f$ *and* $\mathbb{D}_{\text{P}}\, g$ *hold. Then* $\mathbb{D}_{\text{R}}\, (\lambda\nu.\texttt{True})$, $\mathbb{D}_{\text{R}}\, (\lambda\nu.\texttt{False})$, $\mathbb{D}_{\text{R}}\, (\lambda\nu.f_\nu = g_\nu)$, $\mathbb{D}_{\text{R}}\, (\lambda\nu.f_\nu \leq g_\nu)$, $\mathbb{D}_{\text{R}}\, (\lambda\nu.f_\nu < g_\nu)$ *and* $\mathbb{D}_{\text{R}}\, (\lambda\nu.f_\nu \neq g_\nu)$ *hold.*

Proof. For e.g. $\lambda\nu.\, f_\nu < g_\nu$, we first get the witnesses for $w_f : \mathbb{D}_{\text{P}}\, f$ and $w_g : \mathbb{D}_{\text{P}}\, g$ by the projections $p_f := \pi_1(w_f)$ and $p_g := \pi_1(w_g)$. If we denote by ρ the "lift by one renaming" $\rho := \lambda x_i.x_{1+i}$ and then the witness $\dot\exists\big(1 \mathbin{\dot+} x_0 \mathbin{\dot+} \rho\,(p_f) \doteq \rho\,(p_g)\big)$ can be used for $\lambda\nu.\, f_\nu < g_\nu$. ◀

From a *mechanisation point of view,* having to provide explicit witnesses is a painful task and we now describe how it can be almost entirely automated. We use the Coq unification mechanism to analyse a meta-level expression of *Diophantine shape* and reflect it into the corresponding object-level witness of types either $\mathbb{D}_{\text{expr}}$ or $\mathbb{D}_{\text{form}}$ together with the proof that it is an appropriate witness. The following lemma provides a way to process a goal such as $\mathbb{D}_{\text{R}}\, R$ depending on the meta-level syntax of R.

▶ **Lemma 4**. *Let* $R, S : (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ *and* $T : \mathbb{N} \to (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$. *We have the maps:*

1. $\mathbb{D}_{\mathrm{R}}\, R \to \mathbb{D}_{\mathrm{R}}\, S \to \mathbb{D}_{\mathrm{R}}(\lambda\nu. R\,\nu \wedge S\,\nu)$
2. $\mathbb{D}_{\mathrm{R}}\, R \to \mathbb{D}_{\mathrm{R}}\, S \to \mathbb{D}_{\mathrm{R}}(\lambda\nu. R\,\nu \vee S\,\nu)$
3. $\mathbb{D}_{\mathrm{R}}\big(\lambda\nu. T\,(\nu\, x_0)\,(\lambda x_i. \nu\, x_{1+i})\big) \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu. \exists u, T\, u\ \nu\big)$
4. $(\forall \nu, S\,\nu \leftrightarrow R\,\nu) \to \mathbb{D}_{\mathrm{R}}\, R \to \mathbb{D}_{\mathrm{R}}\, S$.

With maps 1–3, we cope with conjunction, disjunction, existential quantification. Atomic or already established Diophantine relations are captured by Propositions 2 and 3 or later established results which are declared as *hints* for Coq proof-search tactics. The map number 4 provides a way to replace $\mathbb{D}_{\mathrm{R}}\, S$ with $\mathbb{D}_{\mathrm{R}}\, R$ once a proof that they are logically equivalent is established. Hence, if S cannot be analysed because it does not currently have a Diophantine shape, it can still be replaced by an equivalent relation R, hopefully better behaved.

2.2 Example of a Mechanised Diophantineness Proof

With the example of the "does not divide" relation $u \nmid v := \neg(\exists k, v = k \times u)$, we describe how to use those results to automate the production of the object-level $\mathbb{D}_{\mathrm{R}}$ witness A of Definition 1 from the meta-level representation of a relation of Diophantine shape.

▶ **Proposition 5**. $\forall f\, g : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}, \mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{R}}\,(\lambda\nu. f_\nu \nmid g_\nu)$.

Proof. $u \nmid v = \neg(\exists k, v = k \times u)$ obviously is not in Diophantine shape. We thus first prove the equivalence $u \nmid v \leftrightarrow u = 0 \wedge v \neq 0 \vee \exists a\, b,\ v = a \times u + b \wedge 0 < b < u$ and this new expression now has a Diophantine shape, relying on the Diophantine shape of Euclidian division. Using this equivalence in combination with map 4 of Lemma 4, we replace the goal $\mathbb{D}_{\mathrm{R}}\,(\lambda\nu. f_\nu \nmid g_\nu)$ with $\mathbb{D}_{\mathrm{R}}\,(\lambda\nu. f_\nu = 0 \wedge g_\nu \neq 0 \vee \exists a\, b,\ g_\nu = a \times f_\nu + b \wedge 0 < b \wedge b < f_\nu)$ and then apply maps 1–3 of Lemma 4 until a shape such as those of Proposition 3 appears. ◀

Once established, we can add the map $\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{R}}\,(\lambda\nu. f_\nu \nmid g_\nu)$ in the Diophantine hint database so that later encountered proof goals $\mathbb{D}_{\mathrm{R}}\,(\lambda\nu. f_\nu \nmid g_\nu)$ can be immediately solved. We implemented the Coq tactic `dio_rel_auto` to automate all this work. Apart from the equivalence for $u \nmid v$ and its proof, which cannot be guessed, the rest is effortless.

The recovery of witnesses of Definition 1 from meta-level syntax is automatic and hidden by the use of the `dio_rel_auto` tactic associated with the ever growing hint database. This way, we can proceed as in e.g. Matiyasevich papers where he just transforms a relation into an equivalent Diophantine shape, accumulating more and more Diophantine shapes on the way. This is a huge simplification over having to program witnesses by hand.

2.3 Exponentiation and Bounded Universal Quantification

For now, we introduce the *elimination of the exponential relation* and then of *bounded universal quantification* as black boxes expressed in the theory of Diophantine relations. However we do contribute implementations for both of these hard results. It is not possible for these two mechanised proofs to be described in detail given the page limit. Nonetheless we postpone some remarks and discussions about these proofs in Section 5.

▶ **Theorem 6** (Exponential). $\forall f\, g\, h,\ \mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{P}}\, h \to \mathbb{D}_{\mathrm{R}}(\lambda\nu. f_\nu = g_\nu^{h_\nu})$.

To prove it, one needs a meta-level Diophantine shape for the exponential relation, *the proof of which is nothing short of extraordinary.* This landmark result is due to Matiyasevich [18], but we have implemented the shorter and more up-to-date proof of [20].[2]

[2] As a side remark, assuming atomic polynomials, the computed witnessing Diophantine formula for $\mathbb{D}_{\mathrm{R}}\big(\lambda\nu.\nu\, x_0 = (\nu\, x_1)^{\nu\, x_2}\big)$ is of size 1689.

▶ **Theorem 7** (Bounded U. Quantification). *For* $f : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ *and* $T : \mathbb{N} \to (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$, *we have a map* $\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu.T\,(\nu\,x_0)\ (\lambda x_i.\nu\,x_{1+i})\big) \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu.\forall u, u < f_\nu \to T\,u\ \nu\big)$.

This map can be compared with map 3 of Lemma 4 and allows to recognise bounded universal quantification as a legitimate Diophantine shape. We have implemented the direct proof of Matiyasevich [19] which does not involve a detour through a model of computation. Notice that the bound f_ν in $\forall u,\, u < f_\nu \to \ldots$ is not constant otherwise the elimination of the quantifier would proceed as a simple reduction to a finitary conjunction.

2.4 Reflexive-Transitive Closure is Diophantine

With these tools – elimination of the exponential relation and of bounded universal quantification – we can show that the reflexive and transitive closure of a Diophantine binary relation is itself Diophantine. We assume a binary relation $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ over natural numbers. The Diophantineness of R can be formalised by assuming that e.g. $\lambda\nu.R\ (\nu\,x_1)\ (\nu\,x_0)$ is a Diophantine relation. We show that i-th iterate of R is Diophantine (where i is non-constant).

▶ **Lemma 8**. *Under hypothesis* $H_R : \mathbb{D}_{\mathrm{R}}\ \big(\lambda\nu.R\ (\nu\,x_1)\ (\nu\,x_0)\big)$, *for any* $f, g, i : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ *we have a map* $\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{P}}\, i \to \mathbb{D}_{\mathrm{R}}\ \big(\lambda\nu.R^{i_\nu}\ f_\nu\ g_\nu\big)$.

Proof. Using Euclidean division, we define the `is_digit` $c\ q\ n\ d$ predicate stating that d is the n-th digit of the base q development of number c, as a Diophantine sentence:

$$\texttt{is\_digit}\ c\ q\ n\ d := d < q \wedge \exists a\, b\, t,\ t = q^n \wedge c = (a.q + d).t + b \wedge b < t$$

Diophantineness which follows from Theorem 6. Then we define the `is_seq` $R\ c\ q\ i$ predicate stating that the first $i+1$ digits of c in base q form an R-chain, again with a Diophantine expression, thanks to H_R and Theorem 7:

$$\texttt{is\_seq}\ R\ c\ q\ i := \forall n,\ n < i \to \exists u\, v,\ \texttt{is\_digit}\ c\ q\ n\ u \wedge \texttt{is\_digit}\ c\ q\ (1+n)\ v \wedge R\ u\ v$$

Then we encode $R^i\ u\ v$ by stating that there exists a (large enough) q and a number c such that the first $i+1$ digits of c in base q form an R-chain starting at u and ending at v:

$$R^i\ u\ v\ \leftrightarrow\ \exists q\, c,\ \texttt{is\_seq}\ R\ c\ q\ i \wedge \texttt{is\_digit}\ c\ q\ 0\ u \wedge \texttt{is\_digit}\ c\ q\ i\ v$$

and this expression is accepted as Diophantine by Lemma 4. ◀

We fill in Lemma 8 in the Diophantine hint database and we derive the Diophantineness of the reflexive-transitive closure as a direct consequence of the equivalence $R^*\ u\ v \leftrightarrow \exists i, R^i\ u\ v$.

▶ **Theorem 9**. *For any binary relation* $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$ *and any* $f, g : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$, *we have the map* $\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{R}}\ \big(\lambda\nu.R\ (\nu\,x_1)\ (\nu\,x_0)\big) \to \mathbb{D}_{\mathrm{R}}\ \big(\lambda\nu.R^*\ f_\nu\ g_\nu\big)$.

3 Elementary Diophantine Constraints

Elementary Diophantine constraints are very simple equations where only one instance of either $\dot{+}$ or $\dot{\times}$ is allowed. We give a direct proof that any Diophantine logic formula is semantically equivalent to the satisfiability of a list of elementary Diophantine constraints.

Starting from two copies of $\mathbb{N}$, one called U with u, v, w ranging over U for existentially quantified variables, and another one $\mathsf{V} = \{x_0, x_1, \ldots\}$ for parameters, we define the type of elementary Diophantine constraints by:

$$c : \mathbb{D}_{\mathrm{cstr}} ::= u \doteq n \mid u \doteq v \mid u \doteq x_i \mid u \doteq v \dot{+} w \mid u \doteq v \dot{\times} w \qquad \text{where } n : \mathbb{N}$$

Notice that these constraints do not have a "real" inductive structure, they are flat and of size either 3 or 5. Given two interpretations, $\varphi : \mathsf{U} \to \mathbb{N}$ for variables and $\nu : \mathsf{V} \to \mathbb{N}$ for parameters, it is trivial to define the semantics $[\![c]\!]^\varphi_\nu : \mathbb{P}$ of a single constraint c of type $\mathbb{D}_{\text{cstr}}$:

$$[\![u \doteq n]\!]^\varphi_\nu := \varphi\, u = n \qquad [\![u \doteq v]\!]^\varphi_\nu := \varphi\, u = \varphi\, v \qquad [\![u \doteq v \dot{+} w]\!]^\varphi_\nu := \varphi\, u = \varphi\, v + \varphi\, w$$
$$[\![u \doteq x_i]\!]^\varphi_\nu := \varphi\, u = \nu\, x_i \qquad [\![u \doteq v \dot{\times} w]\!]^\varphi_\nu := \varphi\, u = \varphi\, v \times \varphi\, w$$

Given a list $l : \mathbb{L}\,\mathbb{D}_{\text{cstr}}$ of constraints, we write $[\![l]\!]^\varphi_\nu$ when all the constraints in l are simultaneously satisfied, i.e. $[\![l]\!]^\varphi_\nu := \forall c,\, c \in l \to [\![c]\!]^\varphi_\nu$. We show the following result:

▶ **Theorem 10**. *For any Diophantine formula $A : \mathbb{D}_{\text{form}}$ one can compute a list of elementary Diophantine constraints $l : \mathbb{L}\,\mathbb{D}_{\text{cstr}}$ such that $\forall \nu : \mathsf{V} \to \mathbb{N},\, [\![A]\!]_\nu \leftrightarrow \exists \varphi : \mathsf{U} \to \mathbb{N},\, [\![l]\!]^\varphi_\nu$.*

I.e. for any given interpretation of parameters ν, $[\![A]\!]_\nu$ holds if and only if the constraints in l are simultaneously satisfiable. Hence any Diophantine logic formula is equivalent to the satisfiability of the conjunction of finitely many elementary Diophantine constraints.

The proof of Theorem 10 spans the rest of this section. We will strengthen the result a bit to be able to get an easy argument by induction on A.

▶ **Definition 11**. *Given a relation $R : (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ and an interval $[u_a, u_{a+n}[\subseteq \mathsf{U}$, a* representation of R in $[u_a, u_{a+n}[$ *is given by:*

1. *a list $\mathcal{E} : \mathbb{L}\,\mathbb{D}_{\text{cstr}}$ of constraints and a reference variable $\mathfrak{r} : \mathsf{U}$;*
2. *proofs that $\mathfrak{r}$ and the (existentially quantified) variables occurring in $\mathcal{E}$ belong to $[u_a, u_{a+n}[$;*
3. *a proof that the constraints in $\mathcal{E}$ are always (simultaneously) satisfiable, i.e. $\forall \nu \exists \varphi\, [\![\mathcal{E}]\!]^\varphi_\nu$;*
4. *a proof that the list $(\mathfrak{r} \doteq 0) :: \mathcal{E}$ is equivalent to R, i.e. $\forall \nu,\, R\ \nu \leftrightarrow \big(\exists \varphi,\, \varphi\, \mathfrak{r} = 0 \wedge [\![\mathcal{E}]\!]^\varphi_\nu\big)$.*

It is obvious that a representation of $\lambda\nu.[\![A]\!]_\nu$ in any interval $[u_a, u_{a+n}[$ is enough to prove Theorem 10 because of item 4 of Definition 11. But actually, computing such a representation is simpler than proving Theorem 10 directly.[3]

▶ **Lemma 12**. *For any $a : \mathbb{N}$ and any $A : \mathbb{D}_{\text{form}}$, one can compute a representation of the relation $\lambda\nu.[\![A]\!]_\nu$ in $[u_a, u_{a+n}[$ for some value $n \leq 4|A|$.*[4]

Proof. We show the result by structural induction on A.

- If A is $p \doteq q$ with $p, q : \mathbb{D}_{\text{expr}}$ then we encode p and q as a directed list of constraints. See Appendix B for a detailed explanation on an example;
- When A is $B \dot{\wedge} C$, we get a representation in $[u_a, u_{a+n_A}[$ by induction. Hence, let $(\mathcal{E}_B, \mathfrak{r}_B)$ be the representation of B in $[u_a, u_{a+n_B}[$. Then, inductively again, let $(\mathcal{E}_C, \mathfrak{r}_C)$ be a representation of C at $[u_{a+n_B}, u_{a+n_B+n_C}[$. We define $\mathfrak{r}_A := u_{a+n_A+n_B}$ and $\mathcal{E}_A := (\mathfrak{r}_A \doteq \mathfrak{r}_B \dot{+} \mathfrak{r}_C) :: \mathcal{E}_B +\!\!+ \mathcal{E}_C$ and then $(\mathcal{E}_A, \mathfrak{r}_A)$ represents $A = B \dot{\wedge} C$ in $[u_a, u_{a+1+n_B+n_C}[$;[5]
- The case of $B \dot{\vee} C$ is similar: simply replace $\mathfrak{r}_A \doteq \mathfrak{r}_B \dot{+} \mathfrak{r}_C$ with $\mathfrak{r}_A \doteq \mathfrak{r}_B \dot{\times} \mathfrak{r}_C$;
- We finish with the case when A is $\dot{\exists}\, B$. Let $(\mathcal{E}_B, \mathfrak{r}_B)$ be a representation of B in $[u_a, u_{a+n_B}[$. Let σ be the substitution mapping parameters in V and defined by $\sigma(x_0) := u_{a+n_B}$ and $\sigma(x_{1+i}) := x_i$; existential variables in U are left unmodified by this substitution. Then $(\sigma(\mathcal{E}_B), \mathfrak{r}_B)$ is a representation of $A = \dot{\exists}\, B$ in $[u_a, u_{a+1+n_B}[$.

This concludes the recursive construction of a representation of $\lambda\nu.[\![A]\!]_\nu$. ◀

[3] Proving Theorem 10 directly involves renamings of existential variables and might produce exponential blow-up in the number of constraints when handled naively.

[4] We denote the size of A with $|A|$. The actual statement in the code is a bit more complicated because we also show that the number of elementary constraints can be bounded by $1 + 3|A|$.

[5] Since the intervals $[u_a, u_{a+n_B}[$ and $[u_{a+n_B}, u_{a+n_B+n_C}[$ are built disjoint, there is no difficulty in merging valuations whereas this usually involves renamings when existential variables are not carefully chosen.

4 Single Diophantine Equations

In this section, we show how a list of elementary Diophantine equations can be simulated by a single identity between two Diophantine polynomials. We use the following well known convexity identity to achieve the reduction, the proof of which can be found in Appendix C.

▶ **Proposition 13**. *Let* $(p_1, q_1), \ldots, (p_n, q_n)$ *be a sequence of pairs in* $\mathbb{N} \times \mathbb{N}$. *Then*

$$\sum_{i=1}^{n} 2p_i q_i = \sum_{i=1}^{n} p_i^2 + q_i^2 \;\leftrightarrow\; p_1 = q_1 \wedge \cdots \wedge p_n = q_n$$

We define Diophantine polynomials similar to the Diophantine expressions $\mathbb{D}_{\text{expr}}$ of Section 2.1 except that we now distinguish the types of bound variables (i.e. U) and of parameters (or free variables) (i.e. V) and that the types U and V are not fixed copies of $\mathbb{N}$ anymore, but type parameters of arbitrary value.

▶ **Definition 14**. *The type of* Diophantine polynomials $\mathbb{D}_{\text{poly}}(\mathsf{U}, \mathsf{V})$ *and the type of* single Diophantine equations $\mathbb{D}_{\text{single}}(\mathsf{U}, \mathsf{V})$ *are defined by:*

$$p, q : \mathbb{D}_{\text{poly}}(\mathsf{U}, \mathsf{V}) ::= u : \mathsf{U} \mid x_i : \mathsf{V} \mid n : \mathbb{N} \mid p \dot{+} q \mid p \dot{\times} q \qquad E : \mathbb{D}_{\text{single}}(\mathsf{U}, \mathsf{V}) ::= p \doteq q$$

For $\varphi : \mathsf{U} \to \mathbb{N}$ and $\nu : \mathsf{V} \to \mathbb{N}$ we define the semantic interpretations of polynomials $[\![p]\!]_\nu^\varphi : \mathbb{N}$ and single Diophantine equations $[\![E]\!]_\nu^\varphi : \mathbb{P}$ in the obvious way.

▶ **Theorem 15**. *For any list* $l : \mathbb{L}\,\mathbb{D}_{\text{cstr}}$ *of elementary Diophantine constraints, one can compute a single Diophantine equation* $E : \mathbb{D}_{\text{single}}(\mathbb{N}, \mathbb{N})$ *such that* $\forall \nu \forall \varphi,\ [\![E]\!]_\nu^\varphi \leftrightarrow [\![l]\!]_\nu^\varphi$.

Proof. We write $l = [p_1 \doteq q_1; \ldots; p_n \doteq q_n]$ and then use Proposition 13. In the code, we moreover show that the size of E is linear in the length of l. If needed, one could also show that the degree of the polynomial is less than 4. ◀

▶ **Corollary 16**. *Let* $R : (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$. *Assuming* $\mathbb{D}_{\text{R}}\, R$, *one can compute a single Diophantine equation* $p \doteq q : \mathbb{D}_{\text{single}}(\mathbb{N}, \mathsf{V})$ *such that* $\forall \nu,\ R\ \nu \leftrightarrow \exists \varphi,\ [\![p]\!]_\nu^\varphi = [\![q]\!]_\nu^\varphi$.

Proof. Direct combination of Definition 1 and Theorems 10 and 15. In the formalisation, we also show that the size of the obtained single Diophantine equation is linearly bounded by the size of the witness formula contained in the proof of $\mathbb{D}_{\text{R}}\, R$. ◀

We have shown that the automation we designed to recognise relations of Diophantine shape entail that these relations are also definable by satisfiability of a single equation between Diophantine polynomials, so these tools are sound w.r.t. a formally restrictive characterisation of Diophantineness. One could argue that the above existential quantifier $\exists \varphi$ encodes infinitely many existential quantifiers but it can easily be replaced by finitely many existential quantifiers over the bound variables that actually occur in p or q.

We prove the following result in Appendix D:

▶ **Proposition 17**. *For any single Diophantine equation* $p \doteq q : \mathbb{D}_{\text{single}}(\mathbb{N}, \mathsf{V})$, *one can compute* $n : \mathbb{N}$ *and a new single Diophantine equation* $p' \doteq q' : \mathbb{D}_{\text{single}}(\mathbb{F}_n, \mathsf{V})$ *such that for any* $\nu : \mathsf{V} \to \mathbb{N}$, $(\exists \varphi : \mathbb{N} \to \mathbb{N},\ [\![p]\!]_\nu^\varphi = [\![q]\!]_\nu^\varphi) \leftrightarrow (\exists \varphi : \mathbb{F}_n \to \mathbb{N},\ [\![p']\!]_\nu^\varphi = [\![q']\!]_\nu^\varphi)$.

5 Remarks on the Implementation of Matiyasevich's Theorems

5.1 Exponential is Diophantine (Theorem 6)

For the admissibility of the exponential relation as Diophantine, we mainly used the description provided in [20] which, among the many options we considered, seemed the shortest. *Matiyasevich's theorem* stating that there is a Diophantine description of the exponential relation $x, y, z \mapsto x = y^z$ is a masterpiece. The proof of [20] proceeds via solutions of Pell's equation $x^2 - bxy + y^2 = 1$ for $b \geq 2$. Using the second order recurrence relation $\alpha_b(-1) = -1$, $\alpha_b(0) = 0$ and $\alpha_b(n+2) = b\alpha_b(n+1) - \alpha_b(n)$ characterised by the following

$$A_b(n) = (B_b)^n \quad \text{with} \quad A_b(n) := \begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix} \quad \text{and} \quad B_b := \begin{pmatrix} b & -1 \\ 1 & 0 \end{pmatrix}$$

square 2×2 matrix equation, one can describe the set of solutions of Pell's equation by $\{(\alpha_b(n), \alpha_b(n+1)) \mid n \in \mathbb{N}\}$. Then, studying the properties of the sequence $n \mapsto \alpha_b(n)$ in $\mathbb{N}$ or $\mathbb{Z}$, one can show that $\alpha_2(n) = n$ and $n \mapsto \alpha_b(n)$ grows exponentially for $b \geq 3$. Studying the properties of the same sequence in $\mathbb{Z}/p\mathbb{Z}$ (for varying values of the modulus p), one can for instance show that $n = \alpha_2(n) \equiv \alpha_b(n) \bmod b-2$, which relates n and $\alpha_b(n)$ modulo $(b-2)$. With various intricate but elementary results,[6] such as e.g. $\alpha_b(k) \mid \alpha_b(m) \leftrightarrow k \mid m$ and $\alpha_b^2(k) \mid \alpha_b(m) \leftrightarrow k\alpha_b(k) \mid m$ (both for $b \geq 2$ and any $k, m \in \mathbb{N}$), one can show that $a, b, c \mapsto 3 < b \wedge a = \alpha_b(c)$ has a Diophantine representation. In our formalisation, we get a Diophantine logic formula of size 490 as a witness (see `dio_rel_alpha_size`).

Once $\alpha_b(n)$ is proven Diophantine, one can recover the exponential relation $x, y, z \mapsto x = y^z$ using the eigenvalue λ of the matrix B_b which satisfies $\lambda^2 - b\lambda - 1 = 0$. By wisely choosing $m = bq - q^2 - 1$, one gets $\lambda \equiv q \bmod m$ and thus, using the corresponding eigenvector, one derives $q\alpha_b(n) - \alpha_b(n-1) \equiv q^n \bmod m$. For a large enough value of m, hence a large enough value[7] of b, this gives a Diophantine representation of q^n. In our code, we get a Diophantine logic formula of size 1689 as a witness (see `dio_rel_expo_size`).

The main libraries which are needed to solve Pell's equation and characterise its solutions are linear algebra (or at least square 2×2 matrices) over commutative rings such as $\mathbb{Z}$ and $\mathbb{Z}/p\mathbb{Z}$, a good library for modular algebra ($\mathbb{Z}/p\mathbb{Z}$), and the binomial theorem over rings. Without the help of the Coq `ring` tactic, such a development would be extremely painful. These libraries are then used again to derive the Diophantine encoding of the exponential.

5.2 Admissibility of Bounded Universal Quantification (Theorem 7)

As explained earlier, we provide an implementation of the algorithm for the elimination of bounded universal quantification described in [19]. It does not involve the use of a model of computation, hence does not create a chicken-and-egg problem when used for the proof of the DPRM theorem. The technique of [19] uses the exponential function and thus Theorem 6 (a lot), and a combination of arithmetic and bitwise operations over $\mathbb{N}$ through base 2 and base 2^q representations of natural numbers.

The Diophantine admissibility of bitwise operations over $\mathbb{N}$ is based on the relation stating that every bit of a is lower or equal than the corresponding bit in b and denoted $a \preccurlyeq b$. The equation $a \preccurlyeq b \leftrightarrow \mathcal{C}_b^a$ is odd (where $\mathcal{C}_b^a$ denotes the binomial coefficient) gives

[6] by elementary we certainly do not mean either simple or obvious, but we mean that they only involve standard tools from modular and linear algebra.

[7] the largeness of which is secured using α itself again, but with other input values.

a Diophantine representation for $a \preccurlyeq b$ and then bitwise operators are derived from $\preccurlyeq$ in combination with regular addition $+$, in particular, the *digit by digit AND* operation called "projection." To obtain that $a \preccurlyeq b$ holds if and only if $\mathcal{C}_b^a \equiv 1 \bmod 2$, we prove Lucas's theorem [17] which allows for the computation of the binomial coefficient in base p. It states that $\mathcal{C}_b^a \equiv \mathcal{C}_{b_n}^{a_n} \times \cdots \times \mathcal{C}_{b_0}^{a_0} \bmod p$ holds when p is prime and $a = a_n p^n + \cdots + a_0$ and $b = b_n p^n + \cdots + b_0$ are the respective base p representations of a and b.[8] A Diophantine representation of the binomial coefficient can be obtained via the binomial theorem: $\mathcal{C}_n^k$ is the k-th digit of the development of $(1+q)^n = \sum_{i=0}^n \mathcal{C}_n^i q^i$ in base $q = 2^{n+1}$. This gives a Diophantine representation using Theorem 6 and the relation `is_digit` defined for Lemma 8.

The rest of the admissibility proof for bounded universal quantification $\forall i,\, i < n \to A$ is a very nice encoding of vectors of natural numbers of type $\mathbb{N}^n$ into natural numbers $\mathbb{N}$ such that regular addition $+$ (resp. multiplication $\times$) somehow performs parallel/simultaneous additions (resp. multiplications) on the encoded vectors. More precisely, a vector $(a_1, \ldots, a_n) \in [0, 2^q - 1]^n$ of natural numbers is encoded as the "cipher" $a_1 r^2 + a_2 r^4 + a_3 r^8 + \cdots + a_n r^{2^n}$ with $r = 2^{4q}$. In these *sparse ciphers,* only the digits occurring at r^{2^i} are non-zero. We remark that none of the parameters, including n or q, are constant in the encoding.

Besides a low-level inductive proof of Lucas's theorem, the essential library for the removal of bounded universal quantification consists in tools to manipulate the type $\mathbb{N}$ simultaneously and smoothly both as (a) usual natural numbers and (b) sparse base $r = 2^{4q}$ encodings of vectors of natural numbers in $[0, 2^q - 1]$. Notice that r is defined as $r = 2^{2q}$ in [19] but we favour the alternative choice $r = 2^{4q}$ which allows for an easier soundness proof for vector multiplication because there is no need to manage for digit overflows (see Appendix E).

A significant step in the Diophantine encoding of $+$ and $\times$ on $\mathbb{N}^n$ is the Diophantine encoding of $u = \sum_{i=1}^n r^{2^i}$ and $u_1 = \sum_{i=2}^{n+1} r^{2^i}$ as the ciphers of the constant vectors $(1, \ldots, 1) \in \mathbb{N}^n$ and $(0, 1, \ldots, 1) \in \mathbb{N}^{n+1}$ respectively, obtained by masking u^2 with $w = \sum_{i=0}^{2^{n+1}} r^i$ and $2w$.

Finally, it should be noted that prior to the elimination of the quantifier in $\forall i,\, i < n \to A$, the Diophantine formula A is first normalised into a conjunction of elementary constraints using Theorem 10, and then the elimination is performed on that list of elementary constraints, encoding e.g. $u \doteq v \dot{+} w$ and $u \doteq v \dot{\times} w$ with their respective sparse cipher counterparts.

6 Minsky Machines Reduce to FRACTRAN

6.1 Minsky Machines

We employ Minsky machines [22] with instructions $\iota : \mathsf{I}_n ::= \texttt{INC}\ (\alpha : \mathbb{F}_n) \mid \texttt{DEC}\ (\alpha : \mathbb{F}_n)\ (p : \mathbb{N})$ as formalised in [10]. A Minsky machine with n registers is a sequence of consecutively indexed instructions $s : \iota_0;\ \ldots\ s+k : \iota_k;$ represented as a pair $(s : \mathbb{N}, [\iota_0; \ldots; \iota_k] : \mathbb{L}\,\mathsf{I}_n)$. Its state $(i, \vec{v})$ is a program counter (PC) value $i : \mathbb{N}$ and a vector of values for registers $\vec{v} : \mathbb{N}^n$. $\texttt{INC}\ \alpha$ increases the value of register α and the PC by one. $\texttt{DEC}\ \alpha\ p$ decreases the value of register α by one if that is possible and increases the PC, or, if the register is already 0, jumps to PC value p. Given a Minsky machine (s, P), we write $(s, P) \mathbin{/\!\!/_M} (i_1, \vec{v}_1) \succ^n (i_2, \vec{v}_2)$ when (s, P) transforms state $(i_1, \vec{v}_1)$ into $(i_2, \vec{v}_2)$ in n steps of computation. For (s, P) to do a step in state $(i, \vec{v})$ the instruction at label i in (s, P) is considered. When a label i is outside of the code of (s, P) we write $\texttt{out}\ i\ (s, P)$ and in that case (and only that case), no computation step can occur. We define the halting problem for Minsky Machines as

$$\mathsf{MM}\big(n : \mathbb{N}, P : \mathbb{L}\,\mathsf{I}_n, \vec{v} : \mathbb{N}^n\big) := (1, P) \mathbin{/\!\!/_M} (1, \vec{v})\!\downarrow$$

[8] With the usual convention that $\mathcal{C}_b^a = 0$ when $a > b$.

where $(s,P)\ /\!\!/_M\ (i,\vec{v})\!\downarrow\ :=\ \exists n\, j\, \vec{w},\ (s,P)\ /\!\!/_M\ (i,\vec{v})\ \succ^n\ (j,\vec{w}) \wedge \texttt{out}\ j\ (s,P)$, meaning that the machine (s,P) has a terminating computation starting at state $(i,\vec{v})$. We refer to [10] for a more in-depth formal description of those counter machines. Note that the halting problem defined there is more specific than the problem MM above defined but both are proved undecidable in our library.

We say that a machine *has a self-loop* if it contains an instruction of the form $i : \texttt{DEC}\ \alpha\ i$, i.e. jumps to itself in case the register α has value 0, leading necessarily to non-termination. For every machine P with self-loops, we can construct an equivalent machine Q using one additional register α_0 with constant value 0, which has the same behaviour but no self-loops. Since the effect of a self loop $i : \texttt{DEC}\ \alpha\ i$ is either decrement and move to the next instruction at $i+1$ if $\alpha > 0$ or else enter in a forever loop at i, it is easily simulated by a jump to a length-2 cycle, i.e. replacing $i : \texttt{DEC}\ \alpha\ i$ with $i : \texttt{DEC}\ \alpha\ j$ and adding $j : \texttt{DEC}\ \alpha_0\ (j+1); j+1 : \texttt{DEC}\ \alpha_0\ j$ somewhere near the end of the program. See Appendix F for the detailed proof of:

▶ **Theorem 18**. *Given a Minsky machine P with n registers one can compute a machine Q with $1+n$ registers and no self loops s.t. for any $\vec{v}$, $(1,P)\ /\!\!/_M\ (1,\vec{v})\!\downarrow\ \leftrightarrow\ (1,Q)\ /\!\!/_M\ (1,0::\vec{v})\!\downarrow$.*

A predicate $R : \mathbb{N}^n \to \mathbb{P}$ *is MM-recognisable* if there exist $m : \mathbb{N}$ and a Minsky machine $P : \mathbb{L}\,\mathsf{I}_{n+m}$ of $(n+m)$ registers such that for any $\vec{v} : \mathbb{N}^n$ we have $R\ \vec{v} \leftrightarrow (1,P)\ /\!\!/_M\ (1, \vec{v} +\!\!+ \vec{0})\!\downarrow$. The last m registers serve as spare registers during the computation. Notice that not allowing for spare registers would make e.g. the empty predicate un-recognisable.[9] It is possible to limit the number of (spare) registers but that question is not essential in our development.

6.2 FRACTRAN

We formalise the language FRACTRAN, introduced as an easy way to describe universal programming language for arithmetic by Conway [2]. A FRACTRAN program Q consists of a list of positive fractions $[p_1/q_1; \ldots; p_n/q_n]$. The current state of a FRACTRAN program is just a natural number s. The state s changes when there is a fraction p_i/q_i s.t. $s.(p_i/q_i)$ is still integral, and in that case only the first one of such fraction is picked up, and $s.(p_i/q_i)$ becomes the new state.

We make this precise inductively for Q being a list of fractions $p/q : \mathbb{N} \times \mathbb{N}$:

$$\frac{q.y = p.x}{(p/q :: Q)\ /\!\!/_F\ x \succ y} \qquad \frac{q \nmid p.x \qquad Q\ /\!\!/_F\ x \succ y}{(p/q :: Q)\ /\!\!/_F\ x \succ y}$$

i.e. at state x the first fraction p/q in Q where q divides $p.x$ is used, and x is multiplied by p and divided by q. For instance, the FRACTRAN program $[5/7; 2/1]$ runs forever when starting from state 7, producing the sequence $5 = 7.(5/7)$, $10 = 5.(2/1)$, $20 = 10.(2/1)$ ...[10]

We say that a FRACTRAN program $Q = [p_1/q_1; \ldots; p_n/q_n]$ is *regular* if none of its denominators is 0, i.e. if $q_1 \neq 0, \ldots, q_n \neq 0$. For a FRACTRAN program $Q : \mathbb{L}\,(\mathbb{N} \times \mathbb{N})$ and $s : \mathbb{N}$, we define the decision problem as the question "does Q halt when starting from s":

$$\mathsf{FRACTRAN}(Q,s) := Q\ /\!\!/_F\ s\!\downarrow \quad \text{with } Q\ /\!\!/_F\ s\!\downarrow\ := \exists x,\ Q\ /\!\!/_F\ s\ \succ^*\ x \wedge \forall y,\ \neg\, Q\ /\!\!/_F\ x \succ y$$

Following [2], we now show how (regular) FRACTRAN halting can be used to simulate Minsky machines halting. The idea is to use a simple Gödel encoding of the states of a Minsky

[9] For any Minsky machine $(1,P)$, if it starts on large enough register values, for instance if they are all greater than the length of P, then no jump can occur and the machine terminates after its last instruction executes. Such unfortunate behavior can be circumvented with a 0-valued spare register.

[10] No FRACTRAN program can ever stop when it contains a fraction having an integer value like $2/1$.

machine. We first fix two infinite sequences of prime numbers $\mathfrak{p}_0, \mathfrak{p}_1, \ldots$ and $\mathfrak{q}_0, \mathfrak{q}_1, \ldots$ all distinct from each other. We define the encoding of n-register Minsky machine states as $\overline{(i, \vec{v})} := \mathfrak{p}_i \mathfrak{q}_0^{x_0} \ldots \mathfrak{q}_{n-1}^{x_{n-1}}$ where $\vec{v} = [x_0, \ldots, x_{n-1}]$:

- To simulate the step semantics of Minsky machines for $i : \texttt{INC}\ \alpha$, we divide the encoded state by $\mathfrak{p}_i$ and multiply by $\mathfrak{p}_{i+1}$ for the change in PC value, and increment the register α by multiplying with $\mathfrak{q}_\alpha$, hence we add the fraction $\mathfrak{p}_{i+1}\mathfrak{q}_\alpha/\mathfrak{p}_i$;
- To simulate $i : \texttt{DEC}\ \alpha\ j$ when $\vec{v}[\alpha] = 1 + n$ we divide by $\mathfrak{p}_i$, multiply by $\mathfrak{p}_{i+1}$ and decrease register α by dividing by $\mathfrak{q}_\alpha$, hence we add the fraction $\mathfrak{p}_{i+1}/\mathfrak{p}_i\mathfrak{q}_\alpha$;
- To simulate $i : \texttt{DEC}\ \alpha\ j$ when $\vec{v}[\alpha] = 0$ we divide by $\mathfrak{p}_i$ and multiply by $\mathfrak{p}_j$. To make sure that this is only executed when the previous rule does not apply, we add the fraction $\mathfrak{p}_j/\mathfrak{p}_i$ *after* the fraction $\mathfrak{p}_{i+1}/\mathfrak{p}_i\mathfrak{q}_\alpha$.

In short, we define the encoding of labelled instructions and then programs as

$$\begin{array}{l} \overline{(i, \texttt{INC}\ \alpha)} := [\mathfrak{p}_{i+1}\mathfrak{q}_\alpha/\mathfrak{p}_i] \\ \overline{(i, \texttt{DEC}\ \alpha\ j)} := [\mathfrak{p}_{i+1}/\mathfrak{p}_i\mathfrak{q}_\alpha; \mathfrak{p}_j/\mathfrak{p}_i] \end{array} \qquad \overline{(i, [\iota_0; \ldots; \iota_k])} := \overline{(i, \iota_0)} +\!\!+ \cdots +\!\!+ \overline{(i+k, \iota_k)}.$$

Notice that we only produce regular programs and that a self loop like $i : \texttt{DEC}\ \alpha\ i$, jumping on itself when $\vec{v}[\alpha] = 0$, will generate the fraction $\mathfrak{p}_i/\mathfrak{p}_i$ potentially capturing any state $\overline{(j, \vec{v})}$ even when $j \neq i$. So this encoding does not work on Minsky machines containing self loops.

▶ **Lemma 19**. *If $(1, P)$ has no self loops then $(1, P) \mathbin{/\!\!/}_M (1, \vec{v})\downarrow \leftrightarrow \overline{(1, P)} \mathbin{/\!\!/}_F \overline{(1, \vec{v})}\downarrow$.*

Proof. Let (i, P) be a Minsky machine with no self loops. We show that the simulation of (i, P) by $\overline{(i, P)}$ is 1-1, i.e. each step is simulated by one step. We first show the forward simulation, i.e. that $(i, P) \mathbin{/\!\!/}_M (i_1, \vec{v}_1) \succ (i_2, \vec{v}_2)$ entails $\overline{(i, P)} \mathbin{/\!\!/}_F \overline{(i_1, \vec{v}_1)} \succ \overline{(i_2, \vec{v}_2)}$, by case analysis. Conversely we show that if $\overline{(i, P)} \mathbin{/\!\!/}_F \overline{(i_1, \vec{v}_1)} \succ st$ holds then $st = \overline{(i_2, \vec{v}_2)}$ for some $(i_2, \vec{v}_2)$ such that $(i, P) \mathbin{/\!\!/}_M (i_1, \vec{v}_1) \succ (i_2, \vec{v}_2)$. Backward simulation involves the totality of MM one step semantics and the determinism of regular FRACTRAN one step semantics combined with the forward simulation.

Using these two simulation results, the desired equivalence follows by induction on the length of terminating computations. ◀

▶ **Theorem 20**. *For any n-register Minsky machine P one can compute a regular FRACTRAN program Q s.t. $(1, P) \mathbin{/\!\!/}_M (1, [x_1; \ldots; x_n])\downarrow \leftrightarrow Q \mathbin{/\!\!/}_F \mathfrak{p}_1\mathfrak{q}_1^{x_1} \ldots \mathfrak{q}_n^{x_n}\downarrow$ holds for any $x_1, \ldots, x_n$.*

Proof. Using Theorem 18, we first compute a Minsky machine $(1, P_1)$ equivalent to $(1, P)$ but with one extra 0-valued spare register and no self loops. Then we apply Lemma 19 to $(1, P_1)$ and let $Q := \overline{(1, P_1)}$. The program Q is obviously regular and given $\vec{v} = [x_1; \ldots; x_n]$, the encoding of the starting state $(1, 0 :: \vec{v})$ for $(1, P_1)$ is $\mathfrak{p}_1\mathfrak{q}_0^0\mathfrak{q}_1^{x_1} \ldots \mathfrak{q}_n^{x_n}$ hence the result. ◀

This gives us a formal constructive proof that (regular) FRACTRAN is Turing complete as a model of computation and is consequently undecidable.

▶ **Corollary 21**. *Halt reduces to FRACTRAN.*

Proof. Theorem 20 gives us a reduction from MM to FRACTRAN which can be combined with the reduction of Halt to PCP from [7] and a slight modification of PCP to MM from [10]. ◀

7 Diophantine Encoding of FRACTRAN

We show that a single step of FRACTRAN computation is a Diophantine relation.

▶ **Lemma 22**. *For any FRACTRAN program* $Q : \mathbb{L}(\mathbb{N} \times \mathbb{N})$ *and any* $f, g : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$, *there is a map* $\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{R}}\, (\lambda\nu.\, Q \mathbin{/\!\!/_F} f_\nu \succ g_\nu)$.

Proof. The map is built by induction on Q. If $Q = [\,]$, then we show $[\,] \mathbin{/\!\!/_F} f_\nu \succ g_\nu \leftrightarrow \mathtt{False}$, and thus $\mathbb{D}_{\mathrm{R}}\, (\lambda\nu.Q \mathbin{/\!\!/_F} f_\nu \succ g_\nu)$ by map 4 of Lemma 4 and Proposition 3. If Q is a composed list $Q = p/q :: Q'$, then we show the equivalence

$$(p/q :: Q') \mathbin{/\!\!/_F} f_\nu \succ g_\nu \;\leftrightarrow\; q.g_\nu = p.f_\nu \vee q \nmid (p.f_\nu) \wedge Q' \mathbin{/\!\!/_F} f_\nu \succ g_\nu$$

and we derive $\mathbb{D}_{\mathrm{R}}\, (\lambda\nu.Q \mathbin{/\!\!/_F} f_\nu \succ g_\nu)$ by map 4 of Lemma 4, Proposition 5 and the induction hypothesis, these last steps being automated by the `dio_rel_auto` tactic. ◀

In addition, the "Q has terminated at x" predicate is Diophantine for any FRACTRAN program Q. The proof is similar to the previous one and postponed to Appendix G.

▶ **Lemma 23**. *For any FRACTRAN program* $Q : \mathbb{L}(\mathbb{N} \times \mathbb{N})$ *and any* $f : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$, *there is a map* $\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{R}}\, (\lambda\nu.\forall y, \neg\, Q \mathbin{/\!\!/_F} f_\nu \succ y)$.

We can now deduce a core result of the paper which states that FRACTRAN programs have Diophantine termination predicates.

▶ **Theorem 24**. *If* Q *is a FRACTRAN program and* $\mathbb{D}_{\mathrm{P}}\, f$ *then* $\mathbb{D}_{\mathrm{R}}\, (\lambda\nu.\, Q \mathbin{/\!\!/_F} f_\nu\!\downarrow)$.

Proof. By definition we have $Q \mathbin{/\!\!/_F} f_\nu\!\downarrow \leftrightarrow \exists x\, (Q \mathbin{/\!\!/_F} f_\nu \succ^* x \wedge \forall y, \neg\, Q \mathbin{/\!\!/_F} x \succ y)$ and hence we obtain the claim using Theorem 9 in conjunction with Lemma 22 and Lemma 23. ◀

We conclude with the undecidability of Hilbert's tenth problem by a reduction chain starting from the Halting problem for single tape Turing machines:

▶ **Theorem 25** (Hilbert's tenth problem). *We have the following reduction chain*

$$\mathit{Halt} \preceq \mathit{PCP} \preceq \mathit{MM} \preceq \mathit{FRACTRAN} \preceq \mathit{DIO\_FORM} \preceq \mathit{DIO\_ELEM} \preceq \mathit{DIO\_SINGLE} \preceq \mathit{H10}$$

and as a consequence, H10 is undecidable.

Proof. The proof combines the previous results like Theorems 20 and 24 and Corollary 16. ◀

8 The Davis-Putnam-Robinson-Matiyasevich Theorem

We conclude the paper with a proof of the DPRM theorem stating that recursively enumerable predicates are Diophantine. Here we assume that the informal notion of "recursive enumerability" can be characterised by Minsky machines recognisability (see Section 6.1).

▶ **Lemma 26**. *For FRACTRAN programs* Q *we have* $\mathbb{D}_{\mathrm{R}}\, \left(\lambda\nu.\, Q \mathbin{/\!\!/_F} \mathfrak{p}_1 \mathfrak{q}_1^{\nu\, x_0} \ldots \mathfrak{q}_n^{\nu\, x_{n-1}}\!\downarrow\right)$.

Proof. By induction on $n : \mathbb{N}$, we show $\forall f,\, \mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{R}}(\lambda\nu.f_\nu = \mathfrak{p}_1 \mathfrak{q}_1^{\nu\, x_0} \ldots \mathfrak{q}_n^{\nu\, x_{n-1}})$. Notice that $\mathfrak{p}_1$ and the $\mathfrak{q}_i$'s are hard-coded[11] in the Diophantine representation but we of course use Theorem 6. Then we end the proof by a combination with Theorem 24. ◀

▶ **Theorem 27** (DPRM). *Any MM-recognisable relation* $R : \mathbb{N}^n \to \mathbb{P}$ *is Diophantine: one can compute a single Diophantine equation* $p \doteq q : \mathbb{D}_{\mathrm{single}}(\mathbb{F}_m, \mathbb{F}_n)$ *with* n *parameters and* m *variables s.t.* $\forall \vec{v} : \mathbb{N}^n,\, R\ \vec{v} \leftrightarrow \exists \vec{w} : \mathbb{N}^m,\, [\![p]\!]_{\vec{v}}^{\vec{w}} = [\![q]\!]_{\vec{v}}^{\vec{w}}$.[12]

[11] Which means we do not need to encode the algorithm that actually computes them.

[12] In the notation $[\![p]\!]_{\vec{v}}^{\vec{w}}$ we abusively identify the vector $\vec{v} : \mathbb{N}^n$ (resp. $\vec{w} : \mathbb{N}^m$) with the valuation $\lambda(i : \mathbb{F}_n).\vec{v}[i]$ (resp. $\lambda(j : \mathbb{F}_m).\vec{w}[j]$) that accesses the components of the vector $\vec{v}$ (resp. $\vec{w}$).

Proof. By definition, $R : \mathbb{N}^n \to \mathbb{P}$ is recognised by some Minsky machine P with $(n+m)$ registers, i.e. $R\ \vec{v} \leftrightarrow (1, P)\ /\!/_M\ (1, \vec{v} +\!\!+ \vec{0})\downarrow$. By Theorem 20, we compute a FRACTRAN program Q s.t. $(1, P)\ /\!/_M\ (1, [v_1; \ldots; v_n; w_1; \ldots; w_m])\downarrow \leftrightarrow Q\ /\!/_F\ \mathfrak{p}_1\mathfrak{q}_1^{v_1} \ldots \mathfrak{q}_n^{v_n}\mathfrak{q}_{n+1}^{w_1} \ldots \mathfrak{q}_{n+m}^{w_m}\downarrow$.

Hence we deduce $R\ [v_1; \ldots; v_n] \leftrightarrow Q\ /\!/_F\ \mathfrak{p}_1\mathfrak{q}_1^{v_1} \ldots \mathfrak{q}_n^{v_n}\downarrow$. As a consequence, the relation $\lambda\nu.R\ [\nu\, x_0; \ldots; \nu\, x_{n-1}]$ is Diophantine by Lemma 26. By Corollary 16, there is a Diophantine equation $p \doteq q : \mathbb{D}_{\text{single}}(\mathbb{N}, \mathsf{V})$ such that $R\ [\nu\, x_0; \ldots; \nu\, x_{n-1}] \leftrightarrow \exists\varphi,\ [\![p]\!]_\nu^\varphi = [\![q]\!]_\nu^\varphi$. Notice that the value $\nu\, x_i$ of any parameter of $p \doteq q$ greater than x_n does not influence solvability.

Now let m be an upper bound of the number of (existentially quantified) variables in $p \doteq q$. We injectively map those variables in $\mathbb{F}_m$ and we project the parameters of $p \doteq q$ onto $\mathbb{F}_n$ by replacing every parameter greater than x_n with the 0 constant. We get a Diophantine equation $p' \doteq q' : \mathbb{D}_{\text{single}}(\mathbb{F}_m, \mathbb{F}_n)$ of which the solvability at $\vec{v}$ is equivalent to $R\ \vec{v}$. ◀

9 Related and Future Work

Regarding formalisations of Hilbert's tenth problem, there are various unfinished and preliminary results in different proof assistants: Carneiro [1] formalises Matiyasevich's theorem (Diophantineness of exponentiation) in Lean, but does not consider computational models or the DPRM theorem. Pąk formalises results regarding Pell's equation [23] and proves that Diophantine sets are closed under union and intersection [24], both as parts of the Mizar Mathematical Library. Stock et al. [27] report on an unfinished formalisation of the DPRM theorem in Isabelle based on [20]. They cover some parts of the proof, but acknowledge for important missing results like Lucas's or "Kummer's theorem" and a "formalisation of a register machine." Moreover, none of the cited reports considers the computability of the reductions involved or the verification of a universal machine in the chosen model of computation yet, one of them being a necessary proof goal for an actual undecidability result in the classical meta-theories of Isabelle/HOL and Mizar.

Regarding undecidability proofs in type theory, Forster, Heiter, and Smolka [7] reduce the halting problem of Turing machines to PCP. Forster and Larchey-Wendling [10] reduce PCP to provability in linear logic via the halting problem of Minsky machines, which we build on. Forster, Kirst and Smolka develop the notion of synthetic undecidability in Coq and prove the undecidability of various notions in first-order logic [8].

In future work, we want to connect our work to the formalisation of the recent simplified undecidability proof for System F inhabitation by Dudenhefner and Rehof [6], which builds on elementary Diophantine constraints. The undecidability of second-order unification shown by Goldfarb [13] is also by reduction from elementary Diophantine constraints. We want to formalise his proof as an addition to our library of undecidable problems.

In the present paper, we prove that every MM-recognisable problem is Diophantine. This result can be extended to an equivalence, and furthermore to other formalised models of computation like μ-recursive functions [16], Turing machines, or the untyped λ-calculus [11].

References

1 Mario Carneiro. A Lean formalization of Matiyasevič's theorem, 2018. arXiv:1802.01795.

2 John H. Conway. *FRACTRAN: A Simple Universal Programming Language for Arithmetic*, pages 4–26. Springer New York, New York, NY, 1987.

3 Martin Davis. Arithmetical problems and recursively enumerable predicates 1. *The Journal of Symbolic Logic*, 18(1):33–41, 1953.

4 Martin Davis and Hilary Putnam. *A computational proof procedure; Axioms for number theory; Research on Hilbert's Tenth Problem.* Air Force Office of Scientific Research, Air Research and Development, 1959.

5 Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential Diophantine equations. *Annals of Mathematics*, pages 425–436, 1961.

6 Andrej Dudenhefner and Jakob Rehof. A Simpler Undecidability Proof for System F Inhabitation. *TYPES 2018*, 2018.

7 Yannick Forster, Edith Heiter, and Gert Smolka. Verification of PCP-Related Computational Reductions in Coq. In *ITP 2018*, pages 253–269. Springer, 2018.

8 Yannick Forster, Dominik Kirst, and Gert Smolka. On Synthetic Undecidability in Coq, with an Application to the Entscheidungsproblem. In *CPP 2019*, pages 38–51, 2019.

9 Yannick Forster and Dominique Larchey-Wendling. Towards a library of formalised undecidable problems in Coq: The undecidability of intuitionistic linear logic. *Workshop on Syntax and Semantics of Low-level Languages, Oxford*, 2018.

10 Yannick Forster and Dominique Larchey-Wendling. Certified Undecidability of Intuitionistic Linear Logic via Binary Stack Machines and Minsky Machines. In *CPP 2019*, pages 104–117. ACM, 2019.

11 Yannick Forster and Gert Smolka. Weak Call-By-Value Lambda Calculus as a Model of Computation in Coq. In *ITP 2018*, pages 189–206. Springer, 2017.

12 Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für mathematik und physik*, 38(1):173–198, 1931.

13 Warren D. Goldfarb. The undecidability of the secondorder unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

14 David Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8(10):437–479, 1902.

15 J. P. Jones and Y. V. Matijasevič. Register Machine Proof of the Theorem on Exponential Diophantine Representation of Enumerable Sets. *J. Symb. Log.*, 49(3):818–829, 1984.

16 Dominique Larchey-Wendling. Typing Total Recursive Functions in Coq. In *ITP 2017*, pages 371–388. Springer, 2017.

17 Edouard Lucas. Théorie des Fonctions Numériques Simplement Périodiques. [Continued]. *American Journal of Mathematics*, 1(3):197–240, 1878.

18 Yuri V. Matijasevič. Enumerable sets are Diophantine. In *Soviet Mathematics: Doklady*, volume 11, pages 354–357, 1970.

19 Yuri V. Matiyasevich. A new technique for obtaining Diophantine representations via elimination of bounded universal quantifiers. *J. Math. Sci.*, 87(1):3228–3233, 1997.

20 Yuri V. Matiyasevich. On Hilbert's Tenth Problem. Expository Lectures 1, Pacific Institute for the Mathematical Sciences, University of Calgary, February 2000. URL: http://www.mathtube.org/sites/default/files/lecture-notes/Matiyasevich.pdf.

21 Yuri V. Matiyasevich. Martin Davis and Hilbert's Tenth Problem. In *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*. Springer, 2016.

22 Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

23 Karol Pąk. The Matiyasevich Theorem. Preliminaries. *Formalized Mathematics*, 25(4):315–322, 2017.

24 Karol Pąk. Diophantine sets. Preliminaries. *Formalized Mathematics*, 26(1):81–90, 2018.

25 Emil L. Post. Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316, 1944.

26 Julia Robinson. Existential definability in arithmetic. *Transactions of the American Mathematical Society*, 72(3):437–449, 1952.

27 Benedikt Stock et al. Hilbert meets Isabelle: Formalisation of the DPRM theorem in Isabelle. *Isabelle Workshop 2018*, 2018. doi:10.29007/3q4s.

28 The Coq Proof Assistant. http://coq.inria.fr, 2019.

A Some numerical Details about the Coq Code Contents

We give a detailed overview of the structure of the code corresponding to the results presented in this paper, and which was contributed to our Coq library of undecidable problems. The following *lines of code (loc)* measurements combine both definitions and proof scripts but do not account for comments. Notice that there are more files in the whole library than those needed to actually cover H10, but here, we only present the latter. In total, we contribute 12k loc to our undecidability project, 4k being additions to its shared libraries as extensions of the Coq standard library.

Concerning the multi-purpose shared libraries in Shared/Libs/DLW/Utils:

- we implemented finitary sums/products (over monoids) up to the binomial theorem (Newton) over non-commutative rings in sums.v and binomial.v for a total of 550 loc;
- we implemented bitwise operations over $\mathbb{N}$, both a lists of bits in bool_list.v and Peano `nat` in bool_nat.v for a total of 1700 loc;
- we implemented many results about Euclidean division and Bézout's identity in gcd.v, prime numbers and their unboundedness in prime.v, and base p representations in power_decomp.v for a total of 1200 loc;
- we implemented miscellaneous libraries for the reification of bounded_quantification.v (120 loc), the Pigeon Hole Principle in php.v (350 loc) and iterations of binary relations in rel_iter.v (230 loc).

Concerning the libraries for Minsky machines and FRACTRAN programs:

- by a slight update to the existing code [10], we proved in mm_comp.v that MM-termination (on any state) is undecidable (10 loc). Both the pre-existing result (undecidability of MM-termination on the zero state) and the new result derive from the correctness of the compiler of binary stack machines into Minsky machines;
- we implemented the removal of self-loops in Minsky machines in mm_no_self.v (340 loc);
- we construct two infinite sequences of primes $\mathfrak{p}_i$ and $\mathfrak{q}_i$ in prime_seq.v (240 loc);
- FRACTRAN definitions and basic results occur in fractran_defs.v (310 loc) and the verified compiler from Minsky machines to FRACTRAN occurs in mm_fractran.v (300 loc);

Concerning the libraries for proving Matiyasevich's theorems:

- we implemented a library for modular arithmetic ($\mathbb{Z}/p\mathbb{Z}$) in Zp.v (920 loc);
- we implemented a library for 2×2-matrix computation including exponentiation and determinants in matrix.v (210 loc);
- we implemented an elementary proof of Lucas's theorem in luca.v (290 loc);
- the solution $\alpha_b(n)$ of Pell's equation and its (modular) arithmetic properties up to a proof of its Diophantineness are in alpha.v (1150 loc);
- from $\alpha_b(n)$, we implement the meta-level Diophantine encoding of the exponential in expo_diophantine.v (150 loc);
- we implement the sparse ciphers used in the Diophantine elimination of bounded universal quantification in cipher.v (1450 loc).

Concerning the object-level Diophantine libraries:

- the definition of Diophantine logic and basic results is in dio_logic.v (450 loc);
- the definition of elementary Diophantine constraints and the reduction from Diophantine logic is in dio_elem.v (580 loc);
- the definition of single Diophantine equations and the reduction from elementary Diophantine constraints is in dio_single.v (350 loc);
- we implement the object-level Diophantine encoding of the exponential relation in dio_expo.v (130 loc); but all the work is done in the previously mentioned libraries;

- the object-level Diophantine encoding of bounded universal quantification spans over dio_binary.v, dio_cipher.v and dio_bounded.v (430 loc);
- we derive the object-level Diophantine encoding of the reflexive-transitive closure in dio_rt_closure.v (40 loc);
- we implement the object-level Diophantine encoding of the FRACTRAN termination predicate in fractran_dio.v (110 loc).

To finish, the main undecidability results and the DPRM:

- the undecidability of Minsky machines is in HALT_MM.v (20 loc);
- the reduction from MM to FRACTRAN is in MM_FRACTRAN.v (50 loc);
- the Diophantine encoding of FRACTRAN termination is in FRACTRAN_DIO.v (70 loc);
- the whole reduction chain leading to the undecidability of H10 is in H10.v (60 loc);
- and the DPRM theorem is in DPRM.v (45 loc).

B Atomic Formulæ as Elementary Constraints (Lemma 12)

We complete the postponed part of the proof of Lemma 12. We compute a representation for an atomic logical formula $p \doteq q : \mathbb{D}_{\text{form}}$ in an interval $[u_a, u_{a+n}[\subseteq \mathsf{U}$.[13] We describe the technique on the example of the atomic formula $x_1 \dot{+} (x_2 \dot{\times} 5) \doteq x_3$ which we represent in the interval $[u_0, u_{10}[$. Let us consider the following definitions:

$$\mathcal{E} := \begin{bmatrix} u_0 \doteq u_1 \dot{+} u_2 \ ; \ u_3 \doteq u_1 \dot{+} u_4 \ ; \ u_3 \doteq u_2 \dot{+} u_9 \ ; \\ u_4 \doteq u_5 \dot{+} u_6 \ ; \ u_5 \doteq x_1 \ ; \ u_6 \doteq u_7 \dot{\times} u_8 \ ; \ u_7 \doteq x_2 \ ; \ u_8 \doteq 5 \ ; \\ u_9 \doteq x_3 \end{bmatrix} \qquad \mathfrak{r} := u_0$$

The second line of $\mathcal{E}$ encodes the expression $x_1 \dot{+} (x_2 \dot{\times} 5)$ in $[u_4, u_9[$, and the third line encodes the expression x_3 in $[u_9, u_{10}[$ in a directed way: the values of the u_i's are uniquely determined by the values of the x_i's and the value of u_4 (resp. u_9) is always the same as the value of $x_1 \dot{+} (x_2 \dot{\times} 5)$ (resp. x_3). By "directed" we mean that the encoding is oriented bottom-up by the syntactic tree of sub-expressions: each variable in $[u_4, u_9[$ (resp. $[u_9, u_{10}[$) encodes a sub-expression of $x_1 \dot{+} (x_2 \dot{\times} 5)$ (resp. x_3) and its value is always the same as the value of the corresponding sub-expression.

The first line encodes the identity sign in $x_1 \dot{+} (x_2 \dot{\times} 5) \doteq x_3$. Indeed, whatever the values of u_4 and u_9, the three constraints of the first line give enough freedom (in the choice of u_1, u_2, u_3) to always be satisfiable (requirement 3 of Definition 11). But when the single constraint $u_0 \doteq 0$ is added (because $\mathfrak{r}$ is u_0), then u_1 and u_2 must evaluate to 0 (because of $u_0 \doteq u_1 \dot{+} u_2$) and then u_3 must have the same value as both u_4 (because of $u_3 \doteq u_1 \dot{+} u_4$) and u_9 (because of $u_3 \doteq u_2 \dot{+} u_9$), hence the identity $x_1 \dot{+} (x_2 \dot{\times} 5) \doteq x_3$ must be satisfied (requirement 4 of Definition 11).

C Proof of a Convexity Identity (Proposition 13)

We give an elementary arithmetic justification of the result, proof which involves none of the high-level tools of mathematical analysis. We first show the statement

$$\text{for any } a, b : \mathbb{N}, \text{ we have } 2ab \leq a^2 + b^2 \text{ and } 2ab = a^2 + b^2 \leftrightarrow a = b \tag{1}$$

[13] The value of a is an input but the value of n is an output.

Assuming without loss of generality that $a \leq b$, we can write $b = a + \delta$ with $\delta \in \mathbb{N}$ and then, for $\bowtie \in \{\leq, =\}$ we have $2ab \bowtie a^2 + b^2 \leftrightarrow 2a^2 + 2a\delta \bowtie a^2 + a^2 + 2a\delta + \delta^2 \leftrightarrow 0 \bowtie \delta^2$ hence the desired result.

Then we proceed with the proof of $\sum_{i=1}^{n} 2p_iq_i = \sum_{i=1}^{n} p_i^2 + q_i^2 \leftrightarrow p_1 = q_1 \wedge \cdots \wedge p_n = q_n$. The *if case* is obvious so we only describe the *only if case*. If there is $u \in [1, n]$ such that $p_u \neq q_u$ then we have $2p_uq_u < p_u^2 + q_u^2$, and $2p_jq_j \leq p_j^2 + q_j^2$ for all the other $j \in [1, n] - \{u\}$, in all cases by Statement (1). Hence we get $\sum_{i=1}^{n} 2p_iq_i < \sum_{i=1}^{n} p_i^2 + q_i^2$ and the identity is not possible. So the only way to get the identity is when $p_u = q_u$ holds for any $i \in [1, n]$. Despite its "classical logic" taste, this argument can easily be transformed into a constructive one by reasoning inductively on n.

D Proof of Proposition 17

We show that any single Diophantine equation $p \doteq q : \mathbb{D}_{\text{single}}(\mathbb{N}, \mathsf{V})$ can be equivalently replaced by another Diophantine equation $p' \doteq q' : \mathbb{D}_{\text{single}}(\mathbb{F}_n, \mathsf{V})$, i.e.

$$(\exists \varphi : \mathbb{N} \to \mathbb{N},\ [\![p]\!]_\nu^\varphi = [\![q]\!]_\nu^\varphi) \leftrightarrow (\exists \varphi : \mathbb{F}_n \to \mathbb{N},\ [\![p']\!]_\nu^\varphi = [\![q']\!]_\nu^\varphi)$$

We choose n greater that the number of bounded variables which occur in either p or q. Then this subset of $\mathbb{N}$ can be injectively mapped into the finite type $\mathbb{F}_n$ and we use such a renaming to compute (p', q'). Remark that the size of (p', q') is the same as the size of (p, q).

E Avoiding Overflows in the Proof of Theorem 7

The section explains why we slightly modified the original proof of the elimination of bounded universal quantification [19] to avoid overflows when multiplying ciphers. Considering Equation (40) of page 3232, we compute the following product of ciphers

$$\sum_{i=1}^{n} a_i r^{2^i} \times \sum_{i=1}^{n} b_i r^{2^i} = \sum_{i=1}^{n} a_i b_i r^{2^{i+1}} + \sum_{1 \leq i < j \leq n} (a_i b_j + a_j b_i) r^{2^i + 2^j}$$

and we remark that $a_ib_j + a_jb_i$ overflows over $r = 2^{2q}$ for e.g. $a_i = a_j = b_i = b_j = 2^q - 1$. This slight overflow makes the implementation of the proof that the right part $\sum_{i<j}(a_ib_j + a_jb_i)r^{2^i+2^j}$ is masked out in Equation (40) significantly harder.

On the other hand, for r alternatively chosen as e.g. $r = 2^{4q}$, the overflow does not occur any more. With this remark, we do not imply that Equation (40) of [19] is incorrect in any way. However, its formal proof is really more complicated when overflows occur and that situation is straightforward to avoid.

F Removing Self Loops in Minsky Machines (proof of Theorem 18)

We explain how any Minsky machine $(1, P)$ with n registers can be transformed into an equivalent one that uses an extra 0 valued spare register $\alpha_0 = 0 \in \mathbb{F}_{1+n}$ and avoids self loops. Let k be the length of P and let P' be the Minsky machine with $1 + n$ registers defined by performing a 1-1 replacement of instructions of $(1, P)$:

- instructions of the form i : `INC` α are replaced by i : `INC` $(1 + \alpha)$;
- self loops i : `DEC` α i are replaced by i : `DEC` $(1 + \alpha)$ $(2 + k)$;
- proper inside jumps i : `DEC` α j for $i \neq j$ and $1 \leq j \leq k$ are replaced by i : `DEC` $(1 + \alpha)$ j;
- and outside jumps i : `DEC` α j for $j = 0 \vee k < j$ are replaced by i : `DEC` $(1 + \alpha)$ 0.

Then we define $Q := P' +\!\!+ [\texttt{DEC}\ \alpha_0\ 0; \texttt{DEC}\ \alpha_0\ (3+k); \texttt{DEC}\ \alpha_0\ (2+k)]$. Notice that P' is immediately followed $\texttt{DEC}\ \alpha_0\ 0$, i.e. by an unconditional jump to 0 (because α_0 has value 0), and that $(1, Q)$ ends with the length-2 cycle composed of $2{+}k : \texttt{DEC}\ \alpha_0\ (3{+}k); 3{+}k : \texttt{DEC}\ \alpha_0\ (2{+}k)$. We show that $(1, Q)$ is a program without self loops (obvious) that satisfies the required simulation equivalence. Indeed, self loops are replaced by jumps to the length-2 cycle that uses the unmodified register α_0 to loop forever. One should just be careful that the outside jumps of $(1, P)$ do not accidentally fall into that cycle and this is why we redirect them all to PC value 0.

G FRACTRAN has halted is a Diophantine Predicate (Lemma 23)

The map $\forall f, \mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{R}}\, (\lambda\nu.\forall y, \neg\, Q \mathbin{/\!\!/\!\!/}_F f_\nu \succ y)$ is built by induction on Q. If $Q = [\,]$, then we show $(\forall y, \neg\, [\,] \mathbin{/\!\!/\!\!/}_F f_\nu \succ y) \leftrightarrow \texttt{True}$, and thus $\mathbb{D}_{\mathrm{R}}\, (\lambda\nu.\forall y, \neg\, Q \mathbin{/\!\!/\!\!/}_F f_\nu \succ y)$ by map 4 of Lemma 4 and Proposition 3. If $Q = p/q :: Q'$, then we show the equivalence $\forall y, \neg\, Q \mathbin{/\!\!/\!\!/}_F f_\nu \succ y \leftrightarrow q \nmid (p.f_\nu) \wedge \forall y, \neg\, Q' \mathbin{/\!\!/\!\!/}_F f_\nu \succ y$ and we get $\mathbb{D}_{\mathrm{R}}\, (\lambda\nu.\forall y, \neg\, Q \mathbin{/\!\!/\!\!/}_F f_\nu \succ y)$ by map 4 of Lemma 4, Proposition 5 and the induction hypothesis.