The DPRM Theorem in Isabelle

Jonas Bayer
Freie Universität Berlin, Arnimallee 2, 14195 Berlin, Germany
jonas.bayer@fu-berlin.de

Marco David
Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
m.david@jacobs-university.de

Abhik Pal
Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
ab.pal@jacobs-university.de

Benedikt Stock
Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
b.stock@jacobs-university.de

Dierk Schleicher
Technische Universität Berlin, Germany
dierk.schleicher@gmx.de

Abstract

Hilbert's 10th problem asks for an algorithm to tell whether or not a given diophantine equation has a solution over the integers. The non-existence of such an algorithm was shown in 1970 by Yuri Matiyasevich. The key step is known as the DPRM theorem: every recursively enumerable set of natural numbers is Diophantine. We present the formalization of Matiyasevich's proof of the DPRM theorem in Isabelle. To represent recursively enumerable sets in equations, we implement and arithmetize register machines. Using several number-theoretic lemmas, we prove that exponentiation has a diophantine representation. Further, we contribute a small library of number-theoretic implementations of binary digit-wise relations. Finally, we discuss and contribute an `is_diophantine` predicate. We expect the complete formalization of the DPRM theorem in the near future; at present it is complete except for a minor gap in the arithmetization proofs of register machines and extending the `is_diophantine` predicate by two binary digit-wise relations.

2012 ACM Subject Classification Theory of computation → Automated reasoning; Theory of computation → Higher order logic

Keywords and phrases DPRM theorem, Hilbert's tenth problem, Diophantine predicates, Register machines, Recursively enumerable sets, Isabelle, Formal verification

Digital Object Identifier 10.4230/LIPIcs.ITP.2019.33

Category Short Paper

Supplement Material Isabelle formalisation: `https://gitlab.com/hilbert-10/dprm`

Acknowledgements We want to thank everyone who has contributed to this project: Deepak Aryal, Bogdan Ciurezu, Yiping Deng, Prabhat Devkota, Simon Dubischar, Malte Haßler, Yufei Liu and Maria Antonia Oprea. Without their involvement, most notably Yufei Liu's implementation of `is_diophantine`, the project would not stand where it is today. Moreover, we would like to express our sincere gratitude to the entire welcoming and supportive Isabelle community. In particular, we are indebted to Mathias Fleury for all his help with Isabelle. Finally, a big thank you Yuri Matiyasevich for inspiring and initiating the project as well as to our supervisor Dierk Schleicher, for motivating us throughout the project, connecting us to many experts in the field, and all his critical feedback.

1 Introduction

The mathematician David Hilbert is well known for the axiomatic method and his *Hilbert program* on a quest to formalize mathematics. While the dawn of the twentieth century did not witness any computers, let alone interactive theorem provers, Hilbert did write a list of 23 problems to direct the international mathematical community. In the tenth problem, he asked for an algorithm to decide any diophantine equation. After the presentation of the problem in 1900, a negative solution was conjectured by Martin Davis in 1950. Yet, the proof that there is no such algorithm was only completed in 1970 by Yuri Matiyasevich, resulting in the Davis-Putnam-Robinson-Matiyasevich theorem.

In an ongoing effort, we formalize the negative solution to Hilbert's tenth problem and in first instance the proof of the DPRM theorem in Isabelle. This paper presents the formalization of Matiyasevich's proof [4] of the DPRM theorem. A core result is a representation of exponentiation in terms of Diophantine equations, obtained from a generalized Fibonacci sequence. Additionally, we implement and arithmetize register machines, Minsky machines in our case. The simulation of their execution in equations allows us to express a recursively enumerably set in equations. Finally, an obvious prerequisite for the formalization is an `is_diophantine` predicate, which we implement and apply to e.g. the exponential relation.

For our formalization of the DPRM theorem, we discuss three conceptual ingredients in Sections 2, 3 and 4: diophantine predicates, the fact that exponentiation is Diophantine and the arithmetization of Minsky machines. They culminate in the DPRM theorem in Section 5 before we provide the overall conclusion in Section 6.

2 Diophantine Predicates

A diophantine polynomial is constructed from addition and multiplication of integer constants as well as natural variables and parameters. Diophantine relations and sets are then defined as follows.

▶ **Definition 1.** *An n-ary predicate $\mathcal{P}$ is called* diophantine *if there exists a diophantine polynomial D, such that a tuple of parameters $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$ satisfies $\mathcal{P}$ if and only if there exist variables $\mathbf{x} = (x_1, \ldots, x_m) \in \mathbb{N}^m$ such that $D(\mathbf{a}, \mathbf{x}) = 0$.*

▶ **Definition 2.** *A set $\mathcal{A} \subset \mathbb{N}^n$ of n-tuples is called* diophantine *if there exists a diophantine predicate $\mathcal{P}$ such that $\mathbf{a} \in \mathcal{A} \iff \mathcal{P}(\mathbf{a})$.*

Examples of diophantine predicates are $\mathcal{P}(a, b) \equiv a \leq b$ or $\mathcal{P}(a, b) \equiv a \mid b$, represented respectively as $\exists x.\, D(a, b, x) = a - b + x = 0$ or $\exists x.\, D(a, b, x) = ax - b = 0$. The sets of all tuples (a, b) satisfying these relations are examples, respectively, of diophantine sets. A third, more surprising, example is that the set of all primes is diophantine, for which a simple diophantine polynomial in 26 variables can be found [4, Section 1.4.1].

It is an elementary fact that conjunctions and disjunctions of diophantine predicates are diophantine. Rather non-trivial is the fact that exponentiation of natural numbers is diophantine as well: for a long time, the opposite was conjectured, and the negative solution to Hilbert's 10th Problem was established once it was established that exponentiation is diophantine. The proof of this assertion constitutes one of the core steps of the proof of DPRM and is presented in the following section. Using exponentiation, we may also access a much more general class of relations and prove that they are diophantine. For a diophantine representation of binomial coefficients in terms of exponential, diophantine equations, we formalize Lucas's theorem. Similarly, one finds that the binary digit-wise relations orthogonality ($a \perp b := \forall k.\, a_k b_k = 0$) and masking ($a \preceq b := \forall k.\, a_k \leq b_k$) are diophantine. From there, digit-wise multiplication (&&, i.e. binary AND) can be expressed as a diophantine relation, too:

$$a \mathbin{\&\&} b = c \iff c \preceq a \wedge c \preceq b \wedge (a - c) \perp (b - c)$$

One might expect that the above relations should be handled easily on a low level, but this was not the case. In fact, they constituted a significant amount of the formalization efforts and required development of new number-theoretic library of utilities to handle natural numbers digit-wise. To demonstrate this, note that most lemmas in this part of the formalization rely on new functions to access the n-th digit in binary or base b representation of a natural number, which did not exist before (in Isabelle).

The implementation of `is_diophantine`. In principle, diophantine predicates and polynomials are straightforwardly implemented and equipped with an `eval` function. However, there are many possibilities to model the details of representing variables and parameters, which show different usability in formal proofs. Due to the non-existence of dependent types in Isabelle, n-tuples of parameters and variables can be implemented either as maps `nat` $\Rightarrow$ `nat` from indices to values (which are eventually zero) or as finite lists. Additionally, one may choose to treat variables and parameters separately, or consider them part of the same map or list. Each of these implementations has its (dis)advantages, and different progress has been made towards formalizing necessary relations using these predicates.

A difficulty common to all approaches is the exchange and relabelling of variables and parameters. This is necessary when proving that a relation which is expressed as a compound diophantine expression (i.e. as conjunctions and disjunctions of smaller diophantine expressions) is diophantine, too – a fact which is usually mentioned only on the side in paper proofs. The most successful approach so far uses an implementation of `is_diophantine` using one `nat` $\Rightarrow$ `nat` map for parameters and variables alike.

However, as the currently developed theory requires much manual work in its proofs, we are developing and experimenting with alternative implementations in parallel in order to bridge the open gaps. In particular, we still need to prove that the aforementioned binary relations, which are used later in the diophantine representation of register machines, are indeed diophantine.

3 Exponentiation is Diophantine

Exponential relations of the type $p = q^r$ and in particular their diophantine representations are the key bridge to connect the notions of *recursively enumerable* and *diophantine* since exponentiation arises in the arithmetization of register machines, as described in the next section.

Following Matiyasevich [4, Section 3], we define a second-order recurrence $\alpha_b(n)$ similar to the Fibonacci numbers to later obtain an expression of q^r in terms of this sequence. In intermediate steps, the proof uses 2×2 matrices to obtain a first-order sequence as well as a diophantine, closed form for the $\alpha_b(n)$. Unless explicitly stated otherwise, all variables are natural numbers and may take values from $\mathbb{N} = \{0, 1, 2, \ldots\}$.

▶ **Definition 3.** *Let $\alpha_b(n)$ denote the unique second-order recurrence of natural numbers parameterized by $b \geq 2$, that satisfies $\alpha_b(0) = 0$ and $\alpha_b(1) = 1$, and for all $n \in \mathbb{N}$*

$$\alpha_b(n+2) = b\alpha_b(n+1) - \alpha_b(n)$$

To follow the rough argument, first note that $\alpha_2(n) = n$ is linear but $b > 2$ implies that $(b-1)^n \leq \alpha_b(n+1) \leq b^n$, i.e. $\alpha_b(n)$ grows exponentially. Then, using various divisibility and congruence relations[1] of the $\alpha_b(n)$, we obtain a diophantine system of 15 equations in 8 variables (in addition to the three parameters a, b, c) for the relation $a = \alpha_b(c)$ given $b > 3$. Combining these two results, with $m = bq - q^2 - 1$, we can then express

$$q^r \equiv q\alpha_b(r) - \alpha_b(r-1) \pmod{m}$$

which is a diophantine representation of exponentiation for $q > 0$ as intended, using the fact that congruence relations have a diophantine representation. Adding the case $q = 0$ and expanding, we obtain the following.

▶ **Theorem 4.** *The predicate $\mathcal{P}_{\text{exp}}(p, q, r) \equiv p = q^r$ has a diophantine representation which is implicitly given by the equivalence to a boolean combination of simpler diophantine relations.*

$$\begin{aligned}\mathcal{P}_{exp}(p, q, r) \iff\ & (q = 0 \land r = 0 \land p = 1) \lor \\ & (q = 0 \land r < 0 \land p = 0) \lor \\ & \exists b, m.\ (b = \alpha_{q+4}(r+1) + q^2 + 2\ \land\ m = bq - q^2 - 1 \\ & \qquad p < m\ \land\ p \equiv q\alpha_b(r) - b\alpha_b(r) + \alpha_b(r+1) \bmod m).\end{aligned}$$

In our contribution, this theorem is fully formalized in rather verbose 2800 loc using 86 intermediate lemmas. Using the currently most successful `is_diophantine` predicate explained above, we then show `is_diophantine` $\mathcal{P}_{\text{exp}}$ in additional 270 loc.

4 Arithmetization of Minsky machines

A core concept in our work are Minsky machines, a type of register machine. We implement them in Isabelle and formally prove their arithmetization, i.e. simulation through equations. Although known to be equivalent to Turing machines, their simpler mode of operations and simpler instructions makes this process easier than for a Turing machine. Every register machine has a finite number of registers R_l and states S_k and is able to execute three types of instructions:

i) S_k: `INC` R_l; S_i
ii) S_k: `DEC` R_l; S_i; S_j
iii) S_k: `HALT`

Each register stores a natural number. In state k, the k-th instruction from the *program* p, i.e. list of instructions, is executed. Instructions of type i) increment some register R_l and move to another state S_i; instructions of type ii) decrement a register R_l and move to some state S_i if the register value was larger than zero, else move to another state S_j; and instructions of type iii) halt execution. In analogy to Turing machines, call the list of register values the *tape* $\mathcal{T}$. A tuple $(k, \mathcal{T})$ is then called a *configuration* of the register machines at some time step t.

At every time step, the current instruction is fetched from the program and the tape is updated accordingly. This way, the next configuration is obtained, until the halt state is reached. With the existing implementation of Turing and Abacus machines by Xu et al. [6]

[1] Matiyasevich notes that these "required properties of numbers $\alpha_b(n)$ can be proved by induction, however, many of them can be made more visual by using matrices." We follow his proof using matrices as given, however an alternative, possibly more direct approach using induction seems feasible, too.

at hand, we modeled our register machines in a fetch-update-step cycle similar to their approach. In addition to being as modular as possible, this hopefully allows more easily for future consolidation of both implementations.

Now, the goal is to find equations which simulate the execution of a register machine. The arithmetization as done by Matiyasevich [4] obtains a set of equations with parameter a which are satisfied if and only if the register machine terminates upon being given a as input in the first register in the initial configuration. In this regard, define the number $r_{l,t}$ to be the value of register l at time t. Similarly, define $s_{k,t}$ to be 1 if the machine is in state k at time t and 0 otherwise. In order to model whether a register has value 0 or not, needed for all decrement states, define *zero indicators* $z_{l,t}$ which are 0 if $r_{l,t} = 0$ and 1 otherwise.

It is straightforward to construct equations for all l, k, t relating all the above numbers to sufficiently and necessarily guard that the program is properly executed and that the machine will halt after a finite number of steps q. However, depending on the input a, q may vary. Should the set of all valid inputs be unbounded, any finite set of equations may not be enough to guarantee termination for all valid inputs. Hence, explicit time-dependence needs to be removed from the equations. This is done by representing the time-evolution of any value in a single natural number, encoded with a sufficiently large basis b, chosen as a power of 2. For example, we accumulate all values of the register l in the number $r_l = \sum_{t=0}^{q} r_{l,t} b^t$. With z_l defined accordingly, the simple inequality $\forall t.\, z_{l,t} \leq 1$ is then encoded as the masking relation $z_l \preceq \sum_{t=0}^{q} 1 \cdot b^t$.

After removal of all explicit time-dependence, only 15 equations remain. We have successfully formalized that these are necessary for an initially valid[2] register machine to terminate in finite time (2400 loc). The much simpler converse statement, the sufficiency of these equations, is almost completely formalized (currently 1100 loc). For its completion, a few more properties of the 15 equations need to be shown; additionally, a few more utilities to work digit-wise with the base b representation of natural numbers need to be developed.

5 All recursively enumerable sets are Diophantine

In a final step, the equations obtained during the arithmetization of register machines need to be proven diophantine. Here, the result of section 3 is again crucial as many exponential relations occur due to the nature of aggregation over time by finite geometric sums as above. The connection to recursively enumerable sets is then readily made as exactly the sets accepted by a register machine are recursively enumerable. Register machines present one instance of an algorithm that can accept the elements of a recursively enumerable set, which is equivalent to having an algorithm that enumerates all elements of the set.

▶ **Definition 5.** *A set $\mathcal{A}$ is* recursively enumerable *if there exists a register machine, i.e. a program p, such that for the initial configuration $(k = 0, \mathcal{T} = [a, 0, \ldots, 0])$, we have $a \in \mathcal{A}$ if and only if the register machine halts after executing p on this configuration for a finite number of steps q.*

Combining all the results of the previous sections, the arithmetization of register machines and the diophantine representation of the resulting equations, including the diophantine representation of exponentiation, we can finally prove and formalize the DPRM theorem.

▶ **Theorem 6** (DPRM). `is_recursively_enumerable` $\mathcal{A} \implies$ `is_diophantine` $\mathcal{A}$

[2] The phrase “initially valid” refers to a set of common-sense validity assumptions about the program and initial configuration, e.g. that all references to registers and states are within bounds, that there is exactly one halt state, etc.

6 Conclusion

Summary of current progress. Our contribution comprises the partial formalization of the proof of the DPRM theorem in Isabelle. This includes an `is_diophantine` predicate for relations and sets, a library of digit-wise operations for natural numbers and corresponding utility functions and lemmas, and an implementation and arithmetization of register (Minsky) machines. The formalization is almost complete, in the sense that the bulk of the proof has been formalized, however two gaps remain. As a minor point, we are yet to complete the proof that the equations obtained from the arithmetization of a register machine are sufficient for the machine to terminate. More importantly, however, we still need to extend the `is_diophantine` predicate and show that binary digit-wise multiplication and binary masking are diophantine relations. Then, we intend to contribute this project to the Isabelle Archive of Formal Proofs (`https://www.isa-afp.org`).

Note that the project is carried out solely by undergraduate students (except the last named author, who is their supervisor and not directly involved in the implementation). They all, including the supervisor, had no prior experience in formalizing proofs. With an overall time span of so far 20 months, this is – to the best of our knowledge – the first major theorem formalized entirely by non-experts in theorem proving. For a more detailed discussion of these aspects of the project, and a reflection of the learning process, please refer to [1].

Related work. Related work on both the DPRM theorem and Hilbert's tenth problem has been carried out in Coq, Mizar and Lean. Larchey-Wendling and Forster [3], working in Coq, recently formalize a clever alternative using Conway's FRACTRAN language to simulate register machines and show undecidability of Hilbert's tenth problem in general. Working in Mizar, Pąk [5] published several articles on formalizing arithmetic properties related to Diophantine equations, notably that exponentiation is diophantine. Carneiro [2], using Pell equations, formalized that exponentiation is diophantine in Lean.

Future outlook. In order to arrive at undecidability of Hilbert's tenth problem from the DPRM theorem, a connection to the undecidability of the Halting problem will need to be made. This requires reference to a specific model of computation, for example our register machines. One possibility is to prove their equivalence to the Abacus or Turing machines formalized by Xu et al. [6] who have previously obtained a suitable undecidability result. Alternatively, the undecidability of our implementation of register machines could be shown directly. Future work may extend this contribution to formalize the whole solution of Hilbert's tenth problem in Isabelle – in the spirit of Hilbert himself.

References

1 Jonas Bayer, Marco David, Abhik Pal, and Benedikt Stock. Beginners' Quest to Formalize Mathematics: A Feasibility Study in Isabelle. In C. Kaliszyk, E. Brady, A. Kohlhase, and C. Sacerdoti Coen, editors, *Conference on Intelligent Computer Mathematics.*, volume 11617 of *Lecture Notes in Computer Science*, 2019. (to appear).

2 Mario Carneiro. A Lean formalization of Matiyasevič's Theorem. `arXiv:1802.01795v1`.

3 Yannick Forster Dominique Larchey-Wendling. Hilbert's Tenth Problem in Coq. URL: `http://www.ps.uni-saarland.de/Publications/documents/Larchey-WendlingForster_2019_H10_in_Coq.pdf`.

4 Yuri Matiyasevich. On Hilbert's Tenth Problem. In Michael Lamoureux, editor, *PIMS Distinguished Chair Lectures*, volume 1. Pacific Institute for the Mathematical Sciences, 2000.

5 Karol Pąk. Progress in the Formalization of Matiyasevich's theorem in the Mizar system. URL: `http://alioth.uwb.edu.pl/~pakkarol/articles/FMM_2018_KP.pdf`.

6 Jian Xu, Xingyuan Zhang, and Christian Urban. Mechanising Turing Machines and Computability Theory in Isabelle/HOL. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving. ITP 2013.*, volume 7998 of *Lecture Notes in Computer Science*, pages 147–162. Springer Berlin, Heidelberg, 2013.