Hilbert's Tenth Problem in Coq

Dominique Larchey-Wendling and Yannick Forster

FSCD 2019
June 25

Introduction

Hilbert's Tenth Problem H10

- Diophantine equation = polynomial eq. over $\mathbb{N}$ (or $\mathbb{Z}$)

$$x^2 + 3z = yz + 2$$

- H10 posed by David Hilbert in 1900:

"Man soll ein Verfahren angeben, nach welchem sich mittels einer endlichen Anzahl von Operationen entscheiden läßt, ob die Gleichung in ganzen Zahlen lösbar ist."

- Essentially asked for a *decision procedure* for solvability of Diophantine equations
- Typical decision problems with a negative answer:
  - does a given Turing machine halt? (Halt)
  - does a given register/Minsky machine halt? (MM)
  - the Post correspondence problem (PCP)
  - is there a proof/term for this formula/type (FOL, syst. F)?

What is so intriguing about H10?

- H10 simple to explain to mathematicians with no CS background
- Hilbert's challenge was hard to solve because of a negative answer:
  - required inventing a formal concept of "decision procedure"
  - algorithms characterized by computability theory (CT, 30-40's)
  - a general notion of computable, and thus non-computable

A short history of H10

1900 Posed by David Hilbert

1944 Emil Post "this begs for an unsolvability proof"

1950s Martin Davis' conjecture: "Every r.e. set is Diophantine"

1953 Davis: "Every r.e. set is Diophantine up to one bounded $\forall$"

1959 Davis and Putnam: "Every r.e. set is exponentially Diophantine"

1961 Julia Robinson: "Every r.e. set is Diophantine if there is at least one Diophantine relation with exponential growth"

1970 Yuri Matiyasevich: "The Fibonacci sequence exhibits exponential growth and is Diophantine."

Resulting in the Davis-Putnam-Robinson-Matiyasevich theorem proving Davis' conjecture.

A library for synthetic undecidability in Coq

`https://github.com/uds-psl/coq-library-undecidability`

Definition (Synthetic undecidability)

P undecidable $:=$ Halting problem reduces to P

- a decision problem $(X, P) : \Sigma(X : \texttt{Type}), X \to \mathbb{P}$
- Many-one reduction from (X, P) to (Y, Q)
  - computable function $f : X \to Y$ s.t. $\forall x,\ P\,x \leftrightarrow Q(f\,x)$
  - "computable" requirement replaced by "defined in CTT"
  - We write $P \preceq Q$ when such reduction exists
- Coq terms are computable (axiom-free)
- Undecidability in Coq by many-one reductions
  - from a seed of undecidability Halt (single tape TM)
  - but also PCP (Forster&Heiter&Smolka, ITP 18)
  - BSM, MM, ILL (Forster&LW, CPP 19)
  - FOL (Forster&Kirst&Smolka, CPP 19) ...

Why add H10 to our library?

- MM halting already in our library (CPP 19)
- Stand-alone reduction from MM (Jones&Matijasevič 84)
  - assuming only Matiyasevich theorem ($z = x^y$ Diophantine)
  - (Matiyasevich 2000) is a very detailed pen&paper proof
- H10 reduces to:
  - system F inhabitation (Dudenhefner&Rehof, TYPES 18)
  - second-order unification (Goldfarb 81)
- H10 allows for easy inter-reducibility proofs
  - enumerating Diophantine solutions is trivial to program
  - an easy way to strengthen Church's thesis
- The DPRM theorem:
  - Diophantine equations can encode any RE-predicate
- Another illustration of capabilities of modern proof assistants

Contribution

First complete mechanisation of H10 and the DPRM theorem

Refactorisation of the proof via FRACTRAN,
easing both explanation and mechanisation

Today

1. Overview of the reduction from Halt to H10, via FRACTRAN
2. Basics of FRACTRAN vs. MM (Conway 87)
3. Details on Diophantine encoding of FRACTRAN
4. H10 and the DPRM theorem
5. Mechanized Diophantine relations
6. Some remarks on the Coq code
7. Related works
8. Overview over the library and future work

Overview of the reduction

From Halt to H10

$$\text{Halt} \preceq \text{MM} \preceq \text{FRACTRAN} \preceq \text{DIO\_*} \preceq \text{H10}$$

- Halt $\preceq$ MM via PCP
  - Halt $\preceq$ PCP via SRS (ITP 18)
  - PCP $\preceq$ MM via Binary Stack Machines (CPP 19)
- MM $\preceq$ FRACTRAN
  - following Conway (87)
  - removing self-loops from MM
- FRACTRAN $\preceq$ DIO_*
  - Diophantine admissibility of RT-closure
  - two results as black-boxes (implemented):
    - Matiyasevich proof (2000) ($z = x^y$)
    - Admissibility of $\forall^{\text{fin}}$ (Matiyasevich 1997)
- Nice factorization of the quite monolithic proof of J&M84

Minsky machines and FRACTRAN

Minsky Machines ($\mathbb{N}$ valued register machines)

Example (transfers α to β in 3 instructions, γ_0 spare register)

$$q : \texttt{DEC}\ \alpha\ (3+q) \qquad q+1 : \texttt{INC}\ \beta \qquad q+2 : \texttt{DEC}\ \gamma_0\ q$$

- n registers of value in $\mathbb{N}$ for a fixed n
- state: $(\texttt{PC}, \vec{v}) \in \mathbb{N} \times \mathbb{N}^n$
- instructions: $\iota ::= \texttt{INC}\ \alpha \mid \texttt{DEC}\ \alpha\ p$
- programs: $(q, [\iota_0; \ldots; \iota_k]) \ \leftrightsquigarrow\ q : \iota_0;\ \ldots\ ;\ q+k : \iota_k$
- Step semantics for INC and DEC (pseudo code)

$$\begin{array}{ll} \texttt{INC}\ \alpha : & \alpha \leftarrow \alpha + 1;\ \texttt{PC} \leftarrow \texttt{PC} + 1 \\ \texttt{DEC}\ \alpha\ p : & \text{if } \alpha = 0 \text{ then } \texttt{PC} \leftarrow p \\ & \text{if } \alpha > 0 \text{ then } \alpha \leftarrow \alpha - 1;\ \texttt{PC} \leftarrow \texttt{PC} + 1 \end{array}$$

- $\boxed{\mathsf{MM}(n, \mathcal{M}, \vec{v}) := (1, \mathcal{M}) /\!/_M (1, \vec{v}) \downarrow}$ (termination in any state)

FRACTRAN (computing with fractions in $\mathbb{Q}^+$)

Example (FRACTRAN program: list of fractions)

$$\left(\frac{455}{33}, \frac{11}{13}, \frac{1}{11}, \frac{3}{7}, \frac{11}{2}, \frac{1}{3}\right)$$

- Program: list of $\mathbb{N} \times \mathbb{N}$; State: a single $x \in \mathbb{N}$
- Step relation is simple to describe
  - pick the first p/q s.t. $x \cdot p/q \in \mathbb{N}$, and this is the new state
  - inductively, characterized by two rules:

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

- Termination predicate

$$Q /\!\!/_F\, s\downarrow := \exists x,\ Q /\!\!/_F\, s \succ^* x \quad \wedge \quad \forall y, \neg(Q /\!\!/_F\, x \succ y)$$

- Decision problem: $\boxed{\mathrm{FRACTRAN}(Q, s) := Q /\!\!/_F\, s\downarrow}$

Conway's reduction from MM to FRACTRAN

- Distinct primes: $\mathfrak{p}_0, \mathfrak{p}_1, \ldots$ and $\mathfrak{q}_0, \mathfrak{q}_1, \ldots$
- Gödel coding of MM-states $\overline{(i, (x_0, \ldots, x_{n-1}))} := \mathfrak{p}_i \mathfrak{q}_0^{x_0} \ldots \mathfrak{q}_{n-1}^{x_{n-1}}$
- Fractional encoding of MM-instructions:

$$\overline{i : \mathtt{INC}\ \alpha} := [\mathfrak{p}_{i+1}\mathfrak{q}_\alpha / \mathfrak{p}_i] \quad \overline{i : \mathtt{DEC}\ \alpha\ j} := [\mathfrak{p}_{i+1}/\mathfrak{p}_i\mathfrak{q}_\alpha ; \mathfrak{p}_j/\mathfrak{p}_i]$$

- and of MM: $\overline{(i, [\iota_0; \ldots; \iota_k])} := \overline{i : \iota_0} +\!\!+ \cdots +\!\!+ \overline{i + k : \iota_k}$
- fails for $i : \mathtt{DEC}\ \alpha\ i$ (self loops) because $\mathfrak{p}_i/\mathfrak{p}_i = 1$
- So first remove self-loops using an extra 0-valued spare variable
  - every MM has an equivalent self-loop free MM
  - self-loops $\rightsquigarrow$ unconditional jump to a length-2 cycle
- Simulate self-loop free MM with FRACTRAN

Theorem (Simulating MM with FRACTRAN)

For any n registers Minsky machine P, one can compute a FRACTRAN program Q s.t. $(1, P) /\!\!/_M (1, [x_1; \ldots; x_n]) \downarrow \leftrightarrow Q /\!\!/_F \mathfrak{p}_1 \mathfrak{q}_1^{x_1} \ldots \mathfrak{q}_n^{x_n} \downarrow$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0: \text{DEC } x_0 \ 2$

$1: \text{DEC } x_1 \ 0$

$2:$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7} \ , \ \frac{5}{2}$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0:$ DEC x_0 2

$1:$ DEC x_1 0

$2:$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{5}{3 \cdot 11}\ ,\ \frac{2}{3}$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

0 : DEC x_0 2
1 : DEC x_1 0
2 :

$$\begin{cases} x_0 = 3 \\ x_1 = 0 \\ \mathsf{PC} = 0 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}\ ,\ \frac{5}{2}\ ,\ \frac{5}{3 \cdot 11}\ ,\ \frac{2}{3}$$

$$s = 2^1 3^0 5^0\ 7^3 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0:$ DEC x_0 2
$1:$ DEC x_1 0
$2:$

$$\begin{cases} x_0 = 2 \\ x_1 = 0 \\ \text{PC} = 1 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}\ ,\ \frac{5}{2}\ ,\ \frac{5}{3 \cdot 11}\ ,\ \frac{2}{3}$$

$$s = 2^0 3^1 5^0\ 7^2 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0 :$ DEC x_0 2
$1 :$ DEC x_1 0
$2 :$

$$\begin{cases} x_0 = 2 \\ x_1 = 0 \\ \mathsf{PC} = 0 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}\ ,\ \frac{5}{2}\ ,\ \frac{5}{3 \cdot 11}\ ,\ \frac{2}{3} \qquad s = 2^1 3^0 5^0\ 7^2 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0:$ DEC x_0 2
$1:$ DEC x_1 0
$2:$

$$\begin{cases} x_0 = 1 \\ x_1 = 0 \\ \mathsf{PC} = 1 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}\ ,\ \frac{5}{2}\ ,\ \frac{5}{3 \cdot 11}\ ,\ \frac{2}{3}$$

$$s = 2^0 3^1 5^0\ 7^1 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0 :$ DEC x_0 2
$1 :$ DEC x_1 0
$2 :$

$$\begin{cases} x_0 = 1 \\ x_1 = 0 \\ \mathsf{PC} = 0 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}\ ,\ \frac{5}{2}\ ,\ \frac{5}{3 \cdot 11}\ ,\ \frac{2}{3}$$

$$s = 2^1 3^0 5^0\ 7^1 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0:$ DEC x_0 2
$1:$ DEC x_1 0
$2:$

$$\begin{cases} x_0 = 0 \\ x_1 = 0 \\ \mathsf{PC} = 1 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}\ ,\ \frac{5}{2}\ ,\ \frac{5}{3 \cdot 11}\ ,\ \frac{2}{3}$$

$$s = 2^0 3^1 5^0\ 7^0 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$$\begin{array}{l} 0 : \text{DEC } x_0 \ 2 \\ 1 : \text{DEC } x_1 \ 0 \\ 2 : \end{array} \qquad \begin{cases} x_0 = 0 \\ x_1 = 0 \\ \text{PC} = 0 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7}, \frac{5}{2}, \frac{5}{3 \cdot 11}, \frac{2}{3} \qquad s = 2^1 3^0 5^0 \ 7^0 11^0$$

Example of MM/FRACTRAN simulation

- A small nullifying MM program
  - two DEC instructions starting at 0 :
  - x_0 is nullified, x_1 zero-valued spare register

$0 :$ DEC x_0 2
$1 :$ DEC x_1 0
$2 :$

$$\begin{cases} x_0 = 0 \\ x_1 = 0 \\ \text{PC} = 2 \end{cases}$$

$$(\mathfrak{p}_0, \mathfrak{p}_1, \mathfrak{p}_2, \ldots) = (2, 3, 5, \ldots) \qquad (\mathfrak{q}_0, \mathfrak{q}_1, \ldots) = (7, 11, \ldots)$$

$$\frac{3}{2 \cdot 7} \;,\; \frac{5}{2} \;,\; \frac{5}{3 \cdot 11} \;,\; \frac{2}{3} \qquad\qquad s = 2^0 3^0 5^1 \; 7^0 11^0$$

FRACTRAN termination is Diophantine

- FRACTRAN step relation is Diophantine:
  - $[] \mathbin{/\!\!/_F} x \succ y \;\leftrightarrow\;$ `False`
  - $p/q :: Q \mathbin{/\!\!/_F} x \succ y \;\leftrightarrow\; q \cdot y = p \cdot x \;\vee\; (q \nmid p \cdot x \;\wedge\; Q \mathbin{/\!\!/_F} x \succ y)$
- FRACTRAN halted at predicate is Diophantine:
  - $\forall y, \neg([] \mathbin{/\!\!/_F} x \succ y) \;\leftrightarrow\;$ `True`
  - $\forall y, \neg(p/q :: Q \mathbin{/\!\!/_F} x \succ y) \;\leftrightarrow\; q \nmid p \cdot x \;\wedge\; \forall y, \neg(Q \mathbin{/\!\!/_F} x \succ y)$
- FRACTRAN halting is Diophantine

$$Q \mathbin{/\!\!/_F} s\downarrow \;\leftrightarrow\; \exists x, \left(Q \mathbin{/\!\!/_F} s \succ^* x\right) \;\wedge\; \forall y, \neg(Q \mathbin{/\!\!/_F} x \succ y)$$

- What closure properties do we need?
  - under polynomial equations (!)
  - under "does not divide" (Euclidean division)
  - under finitary conjunctions and disjunctions, existential quantification
  - under RT-closure (this one is hard!)

Hilbert's Tenth Problem

Theorem (H10)

The solvability of a Diophantine equation is undecidable

- MM halting is undecidable
  - by reduction from Halt via PCP
- FRACTRAN halting is undecidable
  - by reduction from MM
- FRACTRAN halting has a Diophantine representation
  - Given (Q, s) a FRACTRAN program and an initial state
  - compute a polynomial equation which has a solution iff $Q /\!/_F\, s \downarrow$
- a solver for H10 would decide FRACTRAN halting

The DPRM theorem

Theorem (DPRM)

MM-recognisable predicates are Diophantine

- $R : \mathbb{N}^n \to \mathbb{P}$ is recognised by some MM P with $(n + m)$ registers:

$$R\ \vec{v} \leftrightarrow (1, P) /\!\!/_M (1, \vec{v} +\!\!+ \vec{0}) \downarrow$$

- P is equivalent to FRACTRAN Q:

$$(1, P) /\!\!/_M (1, [v_1; \dots; v_n] +\!\!+ \vec{0}) \downarrow \leftrightarrow Q /\!\!/_F \mathfrak{p}_1 \mathfrak{q}_1^{v_1} \dots \mathfrak{q}_n^{v_n} \downarrow$$

- $[s; v_1; \dots; v_n] \mapsto s = \mathfrak{p}_1 \mathfrak{q}_1^{v_1} \dots \mathfrak{q}_n^{v_n}$ is Diophantine
  - by induction on n, using Matiyasevich thm. ($z = x^y$ is Diophantine)
- FRACTRAN halting $s \mapsto Q /\!\!/_F s \downarrow$ is Diophantine

Mechanized Diophantine relations

How to deal smoothly with Diophantine relations

- Diophantine Logic: an expressive language for Diophantine relations
  - not only polynomial equations, but also $\wedge$, $\vee$, $\exists$
  - automated recognition of *Diophantine shapes*
  - possibility to $\boxed{\text{expand}}$ the shapes: $x \nmid y$, $z = x^y$, $\forall^{\text{fin}}$
  - privileged tool for establishing Diophantineness

- Elementary Diophantine constraints:
  - list of $u \doteq n \mid u \doteq v \mid u \doteq x_i \mid u \doteq v \dot{+} w \mid u \doteq v \dot{\times} w$
  - $u, v, w =$ existential variables, $x_i \ldots =$ parameters, $n : \mathbb{N} =$ constant
  - nice intermediate layer, e.g. 2nd-ord. unification or system F

- Single Diophantine Equation: $p \doteq q$
  - p and q are polynomials with variables, constants and parameters
  - H10 is the special case with no parameter

- Conversion from Diophantine Logic $\rightsquigarrow$ Single Diophantine Equation

Diophantine Logic, syntax and semantics (DIO_FORM)

Example (De Bruijn encoding for bound variables)

$$\exists y, (y = 0 \wedge \exists z,\ y = z + 1) \quad \leftrightsquigarrow \quad \dot\exists\big(x_0 \doteq 0 \mathbin{\dot\wedge} \dot\exists(x_1 \doteq x_0 \mathbin{\dot+} 1)\big)$$

- De Bruijn syntax with $\mathsf{V} = \{x_0, x_1, \ldots\} \simeq \mathbb{N}$

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

- Semantics with $\nu : \mathsf{V} \to \mathbb{N}$

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

$$\begin{array}{ll} [\![A \mathbin{\dot\wedge} B]\!]_\nu := [\![A]\!]_\nu \wedge [\![B]\!]_\nu & [\![p \doteq q]\!]_\nu := [\![p]\!]_\nu = [\![q]\!]_\nu \\ [\![A \mathbin{\dot\vee} B]\!]_\nu := [\![A]\!]_\nu \vee [\![B]\!]_\nu & [\![\dot\exists\, A]\!]_\nu := \exists n : \mathbb{N}, [\![A]\!]_{n\cdot\nu} \end{array}$$

with $n{\cdot}\nu\,(x_0) := n$ and $n{\cdot}\nu\,(x_{1+i}) := \nu\, x_i$ (De Bruijn extension)

Diophantine polynomials and relations

Definition (Sub-types of $(\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ and $(\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$)

$$\begin{array}{lll} \mathbb{D}_\mathrm{P}\, f & := \sum p : \mathbb{D}_\mathrm{expr}, \big(\forall \nu, [\![p]\!]_\nu = f_\nu\big) & \text{for } f : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N} \\ \mathbb{D}_\mathrm{R}\, R & := \sum A : \mathbb{D}_\mathrm{form}, \big(\forall \nu, [\![A]\!]_\nu \leftrightarrow R\,\nu\big) & \text{for } R : (\mathsf{V} \to \mathbb{N}) \to \mathbb{P} \end{array}$$

- closure properties for $\mathbb{D}_\mathrm{P}/\mathbb{D}_\mathrm{R}$: provided $\mathbb{D}_\mathrm{P}\, f$ and $\mathbb{D}_\mathrm{P}\, g$ we have

$$\mathbb{D}_\mathrm{P}\,(\lambda\nu.\nu\, x_i) \quad \mathbb{D}_\mathrm{P}\,(\lambda\nu.n) \quad \mathbb{D}_\mathrm{P}\,(\lambda\nu.f_\nu + g_\nu) \quad \mathbb{D}_\mathrm{P}\,(\lambda\nu.f_\nu \times g_\nu)$$

$$\mathbb{D}_\mathrm{R}\,(\lambda\nu.\mathtt{True}) \quad \mathbb{D}_\mathrm{R}\,(\lambda\nu.\mathtt{False}) \quad \mathbb{D}_\mathrm{R}\,(\lambda\nu.f_\nu = g_\nu)$$

$$\mathbb{D}_\mathrm{R}\,(\lambda\nu.f_\nu \leqslant g_\nu) \quad \mathbb{D}_\mathrm{R}\,(\lambda\nu.f_\nu < g_\nu) \quad \mathbb{D}_\mathrm{R}\,(\lambda\nu.f_\nu \neq g_\nu)$$

- for $\mathbb{D}_\mathrm{R}$: for $R, S : (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ and $T : \mathbb{N} \to (\mathsf{V} \to \mathbb{N}) \to \mathbb{P}$ we have

$$\begin{array}{l} \mathbb{D}_\mathrm{R}\, R \to \mathbb{D}_\mathrm{R}\, S \to \mathbb{D}_\mathrm{R}(\lambda\nu.R\,\nu \wedge S\,\nu) \\ \mathbb{D}_\mathrm{R}\, R \to \mathbb{D}_\mathrm{R}\, S \to \mathbb{D}_\mathrm{R}(\lambda\nu.R\,\nu \vee S\,\nu) \\ (\forall \nu, S\,\nu \leftrightarrow R\,\nu) \to \mathbb{D}_\mathrm{R}\, R \to \mathbb{D}_\mathrm{R}\, S \end{array}$$

$$\mathbb{D}_\mathrm{R}\big(\lambda\nu.T\,(\nu\, x_0)\,(\lambda x_i.\nu\, x_{1+i})\big) \to \mathbb{D}_\mathrm{R}\big(\lambda\nu.\exists u, T\,u\,\nu\big)$$

Recognizing more Diophantine shapes

Example (Does not divide is a Diophantine shape)

$$\mathbb{D}_\mathrm{P}\, f \to \mathbb{D}_\mathrm{P}\, g \to \mathbb{D}_\mathrm{R}\,(\lambda\nu.f_\nu \nmid g_\nu)$$

- $f_\nu \nmid g_\nu \leftrightarrow f_\nu = 0 \wedge g_\nu \neq 0 \vee \exists a\, b,\, g_\nu = a \cdot f_\nu + b \wedge 0 < b \wedge b < f_\nu$
- Apply closure properties recursively
- Add the example as *hint for the auto tactic*

Theorem (Exponential (Matiyasevich 1970), proof from (Mat. 2000))

$$\mathbb{D}_\mathrm{P}\, f \to \mathbb{D}_\mathrm{P}\, g \to \mathbb{D}_\mathrm{P}\, h \to \mathbb{D}_\mathrm{R}\big(\lambda\nu.f_\nu = g_\nu^{h_\nu}\big)$$

Theorem ($\forall^{\mathrm{fin}}$, proof from (Matiyasevich 1997))

$$\mathbb{D}_\mathrm{P}\, f \to \mathbb{D}_\mathrm{R}\big(\lambda\nu.\, T\,(\nu\, x_0)\ (\lambda x_i.\nu\, x_{1+i})\big) \to \mathbb{D}_\mathrm{R}\big(\lambda\nu.\forall u, u < f_\nu \to T\ u\ \nu\big)$$

- Add both theorems to the hint database

RT-closure is a Diophantine shape

Theorem (iterations of a binary Diophantine relation)

With $f, g, i : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ *and* $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$

$$\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{P}}\, i \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu.R\ (\nu\, x_1)\ (\nu\, x_0)\big) \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu.R^{i_\nu}\ f_\nu\ g_\nu\big)$$

- Encode R-chains of length i in the digits of c in base q
- $\texttt{is\_d}\ c\ q\ n\ d := d < q \land \exists a\, b,\ c = (a \cdot q + d)\ q^n + b \land b < q^n$
- $\texttt{is\_s}\ R\ c\ q\ i :=$
  $\forall n,\ n < i \to \exists u\, v,\ \texttt{is\_d}\ c\ q\ n\ u \land \texttt{is\_d}\ c\ q\ (1+n)\ v \land R\ u\ v$
- $R^i\ u\ v \ \leftrightarrow\ \exists q\, c,\ \texttt{is\_s}\ R\ c\ q\ i \land \texttt{is\_d}\ c\ q\ 0\ u \land \texttt{is\_d}\ c\ q\ i\ v$

Corollary (reflexive and transitive closure is a Diophantine shape)

With $f, g : (\mathsf{V} \to \mathbb{N}) \to \mathbb{N}$ *and* $R : \mathbb{N} \to \mathbb{N} \to \mathbb{P}$

$$\mathbb{D}_{\mathrm{P}}\, f \to \mathbb{D}_{\mathrm{P}}\, g \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu.R\ (\nu\, x_1)\ (\nu\, x_0)\big) \to \mathbb{D}_{\mathrm{R}}\big(\lambda\nu.R^*\ f_\nu\ g_\nu\big)$$

Elementary Diophantine constraints (DIO_ELEM)

- list of $\mathbb{D}_{\text{cstr}}$: $u \doteq n \mid u \doteq v \mid u \doteq x_i \mid u \doteq v \dot{+} w \mid u \doteq v \dot{\times} w$
- $\varphi : \mathrm{U} \to \mathbb{N}$ for variables and $\nu : \mathrm{V} \to \mathbb{N}$ for parameters

$$\ldots \quad [\![u \doteq x_i]\!]^{\varphi}_{\nu} := \varphi\, u = \nu\, x_i \quad \ldots$$

- representation of $A : \mathbb{D}_{\text{form}}$ into $(\mathfrak{r}, \mathcal{E}) : \mathrm{U} \times \mathbb{L}\, \mathbb{D}_{\text{cstr}}$
  - $\mathcal{E}$ is always satisfiable (for any ν)
  - $(\mathfrak{r} \doteq 0) :: \mathcal{E}$ is satisfiable at ν iff $[\![A]\!]_{\nu}$
  - encode $\dot{\wedge}$ with $\dot{+}$ and $\dot{\vee}$ with $\dot{\times}$
  - encode $\dot{\exists}$ with a De Bruijn extension
  - encode $p \doteq q$ following the syntax tree

Theorem (Diophantine logic to elementary Diophantine constraints)

For $A : \mathbb{D}_{\text{form}}$ one can compute $\mathcal{E} : \mathbb{L}\, \mathbb{D}_{\text{cstr}}$ such that $[\![A]\!]_{\nu} \leftrightarrow \exists \varphi,\ [\![\mathcal{E}]\!]^{\varphi}_{\nu}$

- length of $\mathcal{E}$ linearly bounded by the size of A

Single Diophantine Equation (DIO_SINGLE)

Lemma (Convexity identity)

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

- list of elementary constraints $\rightsquigarrow$ single Diophantine equation
- the size is linear in the length, the degree is at most 4

Theorem (Diophantine relations as polynomial equations)

For any Diophantine relation one can compute an equivalent single Diophantine equation.

- the size is linearly bounded by the size of the witness in $\mathbb{D}_{\text{form}}$
- the degree is at most 4

Code and related works

The Coq code

- included in the library of undecidable problems:

  `https://github.com/uds-psl/coq-library-undecidability`

- also a "frozen" version hyperlinked with the paper:

  `https://uds-psl.github.io/H10`

- devel. of significant size but not unreasonnable
- 12k loc addition to the library
  - 3k loc for Matiyasevich's results ($z = x^y$ and $\forall^{\mathrm{fin}}$)
  - 5k loc the Diophantine, FRACTRAN, H10 and the DPRM
  - 4k loc addition to shared libs
- automation in Diophantineness proofs helped a lot
  - expanding Diophantine shape hints as they get proved

Related work

- Matiyasevich theorem in Lean (Carneiro 2018)
  - no link with computational models
- results about Pell's equation in Mizar (Pak 2017)
  - some basic results about Diophantine relations (Pak 2018)
- the DPRM in Isabelle (Stock et al. 2018-...)
  - still unfinished: `https://gitlab.com/hilbert-10/dprm`

Features of interactive proof assistants used

1. Interactive construction of (computable) functions in proof scripts
2. Basic automation providing proof search using hints
3. Automation for goals involving numbers over rings

Conclusion

Contributions and future work