View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Elsevier - Publisher Connector

ELSEVIER
Theoretical Computer Science 193 (1998) 129–148
Theoretical Computer Science

Rudimentary relations and primitive recursion: A toolbox

Henri-Alex Esbelin[a,*], Malika More[b]

[a] *LLAIC1, IUT Clermont 1, BP 86, 63172 Aubière Cedex, France*
[b] *Département de Mathématiques, Université de Caen, 14032 Caen, France*

Received October 1996; revised December 1996
Communicated by M. Nivat

Abstract

Rudimentary relations are those relations over natural integers that are defined by a first-order arithmetical formula, in which all quantifications are bounded by some variables. The question of whether a given primitive recursive relation is rudimentary is in some cases difficult and related to several well-known open questions in theoretical computer science. In this paper, we present systematic tools to study this question, and various applications. One of them gives a sufficient condition of the collapsing of the first classes of the Grzegorczyk's hierarchy.

Keywords: Rudimentary relations, counting, primitive recursion, coding

1. Introduction

Rudimentary relations are those relations over integers which are definable by a Δ_0-formula. They have been studied for a long time (see [1, 9, 20, 24]) and are still interesting because they form the following *robust*, *large* and *intriguing* class of relations.

Definition 1.1. Let us denote by $\mathfrak{R}$ the smallest class of relations over integers containing the graphs of addition and multiplication (seen as ternary relations) and closed under the following operations:

- boolean operations $(\neg, \wedge, \vee, \rightarrow)$;
- explicit transformations, i.e. adding, cancelling, renaming, permuting and confusing variables, (see a precise definition in [24]);
- variable bounded quantifications (i.e. $\forall x < y \ldots$ meaning $\forall x\ (x < y \rightarrow \ldots)$ and $\exists x < y \ldots$ meaning $\exists x\ (x < y \wedge \ldots)$.

$\mathfrak{R}$ is *robust*: there are several different definitions of this class of relations. Rudimentary relations were first introduced by Smullyan in [24], following the ideas of Quine

* Corresponding author. E-mail: esbelin@llaic.univ-bpclermont.fr.

0304-3975/98/$19.00 © 1998—Elsevier Science B.V. All rights reserved
PII S0304-3975(97)00002-9

(cf. [20]) using dyadic concatenation $x = y * z$ as a basic relation for arithmetic instead of $x = y + z$ and $x = y.z$. In the same paper, Smullyan shows that any semi-recursive relation can be defined by a formula of type as $\exists \boldsymbol{u} P(\boldsymbol{x}, \boldsymbol{u})$, where P is a rudimentary relation, and that all rudimentary relations are definable by Δ_0-formulas. Then, Bennett shows in [1] that the class of rudimentary relations does not depend on what (fixed) alphabet is used to represent integers, as long as it contains at least two letters, and that all Δ_0-formulas define rudimentary relations.

In [9], Harrow proved that $\mathfrak{R}$ is closed under substitution of a polynomial to a variable. In particular, this means we can make use of *polynomially* bounded quantifications (such as $\forall x < y^2$) instead of bounded variable quantifications.

Also, $\mathfrak{R}$ corresponds to a computational complexity class, to a descriptive complexity class and to a weak recursion class as we recall below.

$\mathfrak{R}$ is *large*: most natural arithmetical relations are rudimentary. For instance, the following formula defines the set of prime numbers:

$$x > 1 \wedge \forall y < x \forall z < x \neg (x = y.z).$$

In Section 2, we list some other relations that are proved to be rudimentary. In some cases, we use the fact that the ternary relation $x = y^z$ is rudimentary, as Bennett has proved in [1].

Mainly there are two types of relations whose status toward rudimentary relations is unknown. The first type corresponds to graphs of recursive primitive functions, in particular to counting relations. For instance, *is the binary relation "y is the xth prime number" rudimentary*? Indeed, the question of whether $\mathfrak{R}$ is closed under counting is still open (see [13, 19] and Section 7). In this paper, we are concerned in this first type of relations.

The second type corresponds to relations obtained from rudimentary relations by substituting an exponential to a variable. For instance, *is the unary relation "x verifies* $2^x + 1$ *is prime" rudimentary*? Note that it is known that $\mathfrak{R}$ is not closed under substitution of an exponential to a variable (see [9]), so that the answer is "no" in general.

However, it is difficult to exhibit a *natural* arithmetical relation which can be proved not to be rudimentary.

The origin of this paper is a previous proof of the fact that the sequence of Fibonacci's numbers is rudimentary (see [6, 16] and Section 6). With this aim in view, we had to use various coding devices which are presented in this paper. This paper is an attempt to systematize the use of these tools for proving that various primitive recursive relations are rudimentary (or counting rudimentary, see Section 7).

$\mathfrak{R}$ is *intriguing*: rudimentary relations are linked with a lot of well-known open questions in computational complexity, finite model theory, weak arithmetics and recursivity theory. Let us denote by *RUD* the class of rudimentary sets (i.e. unary rudimentary relations) and by $\mathfrak{L}_2(RUD)$ the class of their dyadic representations ("rudimentary languages"). Wrathall proved that reasonable encoding of k-ary

rudimentary relations into languages provides rudimentary languages (see [27]), so that whenever it is convenient we confuse the two notions.

In *computational complexity theory*, $\mathfrak{L}_2(RUD)$ is proved to be equal to the linear hierarchy *LinH* defined in [27], and consequently that $\mathfrak{L}_2(RUD)$ contains *NLINTIME*. Nepomnjascii proved that $\mathfrak{L}_2(RUD)$ contains *LOGSPACE*, Myhil proved that $\mathfrak{L}_2(RUD)$ is contained in *LINSPACE* (see [18, 21]), hence $\mathfrak{L}_2(RUD)$ is also contained in *NEXPTIME, but none of these inclusions is known to be strict.* Also, it is known that $\mathfrak{L}_2(RUD)$ contains many NP-complete problems, namely all the classical problems listed by Karp in [12] (see also [17]).

In *descriptive complexity*, *RUD* is equal to a class of binary spectra (see [26]). The question whether *RUD contains all binary spectra* is still open, and a positive answer would imply $\mathrm{NP} \neq co - \mathrm{NP}$. On the other hand, (second-order) spectra are represented in rudimentary sets up to some exponential functions (see [17]). Also, concerning *finite model theory*, it is proved that $\mathfrak{L}_2(RUD)$ corresponds to monadic second-order logic with addition (see [17]).

In *formal languages theory*, $\mathfrak{L}_2(RUD)$ is proved to be the smallest class of languages containing the regular languages and $\{1^n 2^n,\ n \geqslant 0\}$ and closed under boolean operations, inverse homomorphisms and length-preserving homomorphisms (see [27]). Jones proved that $\mathfrak{L}_2(RUD)$ strictly contains context-free languages (see [11]) and Myhil proved it is contained into context-sensitive languages (see [18]), but none of these inclusions is known to be strict.

Finally, let us turn to *recursion theory*, which concerns this paper. First, let us recall that $\mathfrak{R}$ is equal to the class of relations associated to the following class of primitive recursive functions G^2 (see [10]): the smallest class of functions containing projections, constants, successor, substraction, product and closed under composition and limited minimum ($f(\boldsymbol{u}) = \mathrm{MIN}\{x < j(\boldsymbol{u}) : g(x, \boldsymbol{u}) = 0\}$ if this set is not empty, and $f(\boldsymbol{u}) = j(\boldsymbol{u})$ otherwise). However, this characterization is not totally satisfying, because limited minimum is not a classical recursive schema.

Let us now recall the definition of the Grzegorczyk's hierarchy for primitive recursive functions (see [8]): for all $i \geqslant 0$, we denote $\mathfrak{E}^i$ the smallest class of functions containing projections, constants and the ith Ackermann's function and closed under composition and bounded recursion ($f(\boldsymbol{x}, 0) = g(\boldsymbol{x}), f(\boldsymbol{x}, i+1) = h(\boldsymbol{x}, i, f(\boldsymbol{x}, i)), f(\boldsymbol{x}, i) < j(\boldsymbol{x}, i)$). This hierarchy is strict and its union is equal to the class of all primitive recursive functions. Let us denote by $\mathfrak{E}^i_*$ the corresponding class of relations, i.e. the relations which characteristic functions are in $\mathfrak{E}^i$. Grzegorczyk proved that the corresponding hierarchy is strict for $i \geqslant 3$ (see [8]) and Ritchie proved it for $i = 2$ (see [21]), but *the case of small classes* $\mathfrak{E}^0_*$ *to* $\mathfrak{E}^2_*$ *is still open* (see [13]). The union of this hierarchy is equal to the class of all primitive recursive relations. It is well known that $\mathfrak{R} \subseteq \mathfrak{E}^0_*$, but *the equality remains an open question* (and would imply $\mathfrak{R} = \mathfrak{E}^2_*$ as well). Note that addition as a function lies in $\mathfrak{E}^1 \setminus \mathfrak{C}^0$, whereas as a relation, it lies in $\mathfrak{E}^0_*$. Similarly, multiplication (resp. exponentiation) as a function lies in $\mathfrak{E}^2 \setminus \mathfrak{E}^1$ (resp. $\mathfrak{E}^3 \setminus \mathfrak{E}^2$) whereas they lie in $\mathfrak{E}^0_*$ as a relation. We shall see in Section 3 that the graphs of all Ackermann's unary functions lie in $\mathfrak{R}$, hence in $\mathfrak{E}^0_*$. Hence, the main

way of exhibiting a primitive recursive relation which is not rudimentary is to choose it in $\mathfrak{E}^3_* \setminus \mathfrak{E}^2_*$. Although it is true that infinitly many relations exist, we know no natural example.

The first author attempts to characterize $\mathfrak{R}$ as a recursion class in [5]. Let $\mathfrak{J}^{-1}$ be the smallest class of functions containing constants, projections and predecessor and closed under composition and bounded iteration ($f(\boldsymbol{x},0)=g(\boldsymbol{x}), f(\boldsymbol{x},i+1)=h(\boldsymbol{x},f(\boldsymbol{x},i))$, $f(\boldsymbol{x},i)<j(\boldsymbol{x},i)$). The corresponding class of relations $\mathfrak{J}^{-1}_*$ contains $\mathfrak{R}$ whereas any class obtained by cancelling some basic operations or by replacing bounded iteration by bounded pure iteration is strictly contained in $\mathfrak{R}$. But, once again, *the question of equality remains open.*

Hence, given a primitive recursive relation, the question of its belonging to $\mathfrak{R}$ is nontrivial, and methodic tools for studying this type of questions are worth being developed.

In Section 2, we write down several easy rudimentary definitions that are used in the sequel. In Section 3, we present the classical encoding of calculus for a primitive recursive function, and study in which case this tool is strong enough to prove that the graph of a given (primitive recursive) function is rudimentary. This method was used by Paris and Wilkie in [19], and by Woods in [26]: his result corresponds to our Lemma 3.6, and proves that some "short and small" recursively defined sequences of integers are rudimentary. He extended this lemma to larger and longer sequences, but only when they are defined by *summation*. We generalize it to larger and longer and *recursively defined* sequences. Then we can obtain results, (some of these have already been proved by Meloul in [15] via LOGSPACE machines), in a straightforward arithmetical way. Then we prove that linear and polynomial recurrences are rudimentary in Section 6. Finally, in Section 7, we prove that the set of the counting rudimentary relations is closed under polynomial substitution, and to obtain a sufficient condition of the collapsing of the Gregorczyk's hierarchy. Sections 6 and 7 can be read independently.

2. Easy rudimentary relations

It will be convenient for the last section to generalize the results to classes which contains the rudimentary relations. So let us introduce the following definition:

Definition 2.1. Let us denote by $\mathfrak{R}^{vb}$ any class of primitive recursive relations over integers containing the graph of addition and multiplication and closed under the following operations:

- boolean operations;
- explicit transformations;
- variables bounded quantifications.

It clearly appears that the class $\mathfrak{R}$ is the smallest among the $\mathfrak{R}^{vb}$ classes.

First, let us consider closure properties of the class $\mathfrak{R}^{vb}$. The following lemma is obvious:

Lemma 2.2. *Let $R(x_1,\ldots,x_n)$ be a relation in $\mathfrak{R}^{vb}$. Let f be a function* (1) *whose graph is in $\mathfrak{R}^{vb}$,* (2) *less than some projection (for all but a finite number of values of the variables). Then the relation $R(x_1,\ldots,x_{l-1},f(x_1,\ldots,x_n),x_{l+1},\ldots,x_n)$ still lies in $\mathfrak{R}^{vb}$.*

As a consequence, it is possible to use within $\mathfrak{R}^{vb}$ quantifications which are bounded by a function satisfying hypothesis of this lemma. The following theorem is due to Harrow (cf. [9]). It generalizes the previous lemma to a polynomially bounded function, in case of its graph is in $\mathfrak{R}$. The theorem 7.4 of this paper establishes the corresponding result for the class $\mathfrak{R}^{\sharp}$.

Theorem 2.3. *Let $R(x_1,\ldots,x_n)$ be a relation in $\mathfrak{R}$. Let f be a function* (1) *whose graph is in $\mathfrak{R}$,* (2) *less than a polynomial (for all but a finite number of values of the variables). Then the relation $R(x_1,\ldots,x_{l-1},f(x_1,\ldots,x_n),\ x_{l+1},\ldots,x_n)$ still lies in $\mathfrak{R}$.*

As a consequence, the graphs of polynomials are in $\mathfrak{R}$ and it is possible to use quantifications bounded by a function satisfying hypotheses of this theorem.

2.1. Relations easily defined without exponentiation

The following relations are easily seen to be rudimentary ones.
$z = rm(x,D)$ (or in some cases, for typographical convenience, $z = rm_{\mathrm{mod}\,D}(x)$) iff
$((z<D)\wedge(\exists q<x+1\ (x=D.q+z))\wedge D>1)\vee(D=1\wedge z=0)$;
$x\equiv y(\mathrm{mod}\,D)$ iff $\exists z<D(z=rm_{\mathrm{mod}\,D}(x))\wedge(z=rm_{\mathrm{mod}\,D}(y))$;
$PRIME(x)$ iff $\neg(x=0)\wedge\neg(x=1)\wedge\forall y<x\forall z<x\neg(x=y.z)$;
$x|y$ iff y divides x;
$SAMEPRIME(x,y)$ iff x and y have the same prime divisors;
$SQUAREFREE(x)$ iff x is divisible by the square of no prime number;
$MINPRIME(x,p)$ iff p is the smallest prime divisor of x;
$SUCCPRIME(x,p,q)$ iff p and q are consecutive prime divisors of x.

Now, let us consider now the function $DIVPRIME(x,i)$ providing the prime divisor of x of rank i. Its graph $p = DIVPRIME(x,i)$ was rudimentary defined as follows by Woods in [26]:

$$
\begin{aligned}
&x\geqslant 2\wedge PRIME(p)\wedge p|x\wedge i<p\wedge\exists y<x+1\,\exists t<y\\
&[SAMEPRIME(x,y)\wedge SQUAREFREE(y)\\
&\quad\wedge\forall q<y+1(MINPRIME(y,q)\rightarrow t\equiv 0(\mathrm{mod}\,q))\wedge\forall q<y\forall q'<y\forall\alpha<q\forall\beta<q'\\
&\quad((SUCCPRIME(y,q,q')\wedge t\equiv\alpha(\mathrm{mod}\,q)\wedge\ t\equiv\beta(\mathrm{mod}\,q'))\rightarrow\beta=\alpha+1)\\
&\quad\wedge t\equiv i(\mathrm{mod}\,p)].
\end{aligned}
$$

The existence of t is ensured by the following theorem.

Theorem 2.4. *Any congruence system* $x \equiv a_i \pmod{b_i}$ *for* $i = 1$ *to* n, *where the* b_i*'s are pairwise coprime admits infinitely many solutions in* $\mathbb{N}$. *Moreover, there is a single solution to the system in* $[0, \prod_{i=1}^{n} b_i[$.

We do not give any proof of the (generally easy) number theoretic results that we use. Except stated otherwise, the reader is invited to refer to the book of Shapiro (cf. [23]).

Now, let us define *CARDDIVPRIME*, which is the following counting relation: $s = CARDDIVPRIME(x)$ iff $s = \#\{p \leqslant x, PRIME(p) \wedge p \mid x\}$. The rudimentarity of this relation follows from the equivalence between $s = CARDDIVPRIME(x)$ and

$$\exists p \leqslant x(p = DIVPRIME(x, s-1) \wedge \neg(\exists q \leqslant x(SUCCPRIME(x, p, q)).$$

2.2. Relations defined with exponentiation

Let us recall a result due to Bennett in [1]. Paris and Dimitracopoulos gave a shorter arithmetical proof in [3].

Theorem 2.5. *The ternary relation* $z = x^y$ *is rudimentary.*

Now, let us turn to frequently used relations:

Let $VAL(x, i) = \alpha_i$ be the valuation of the prime divisor of x of rank i.
The rudimentarity of this relation follows from the equivalence between $\alpha = VAL(x, i)$ and

$$(\exists \beta)_{<x+1}(\exists p)_{<x+1}(\exists q)_{<x+1}(\exists r)_{<x^2+1}$$
$$(\beta = \alpha + 1 \wedge p = DIV(x, i) \wedge q = p^{\alpha} \wedge r = p^{\beta} \wedge q \mid x \wedge \neg(r \mid x)).$$

Let $t = DIGIT(x, y, z)$ give the digit t of weight z^y in the z-expansion of x (with $z \geqslant 2$). This 4-ary relation is rudimentary as the reader can easily verify.

3. History of the calculus

3.1. Coding lemma

All along this paper, by *function* we mean a function from some $\mathbb{N}^k$ to $\mathbb{N}$.

Definition 3.1. The function $f(\boldsymbol{x}, y)$ is defined from the functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x}, y, z)$ by a primitive recursive schema if $\begin{cases} f(\boldsymbol{x}, 0) = g(\boldsymbol{x}) \\ f(\boldsymbol{x}, i+1) = h(\boldsymbol{x}, i, f(\boldsymbol{x}, i)) \end{cases}$.

In this case, g is called the *initial value* of the function, i the *index of recursion* and h the *transition function*. The n-tuple of variables $\boldsymbol{x}$ consists of n *parameters*.

Let $M(\boldsymbol{x}, y)$ denote any number strictly greater than $f(\boldsymbol{x}, i)$ for $i = 0$ to y. The classical *encoding of* (*the history of*) *the calculus* of $f(\boldsymbol{x}, y)$ is $c = \sum_{i=0}^{y} f(\boldsymbol{x}, i) M(\boldsymbol{x}, y)^i$.

Let us notice that we have $c < M(\boldsymbol{x}, y)^{y+1}$. This code is then used according to the following

Proposition 3.2. *We have $z = f(\boldsymbol{x}, y)$ if and only if there exists an integer $c < M(\boldsymbol{x}, y)^{y+1}$ such that for the $M(\boldsymbol{x}, y)$-expansion of c, the following properties hold:*

1. *the digit of weight M^0 of c is $g(\boldsymbol{x})$;*
2. *the digit of weight M^y of c is z;*
3. *if v is the digit of weight M^i of c with $0 \leqslant i < y$, then the digit of weight M^{i+1} of c is $h(\boldsymbol{x}, i, v)$.*

Now, let us turn to the case of rudimentary relations (see [4, 19, 26]). We provide an arithmetical formula denoted by $CODE^f(z, \boldsymbol{x}, y, c, M)$ precisely expressing the fact that the truncature of the M-expansion of c of length y encodes the calculus of $f(\boldsymbol{x}, y)$ with result z, namely $CODE^f(z, \boldsymbol{x}, y, c, M)$ holds if and only if

$$DIGIT(c, 0, M) = g(\boldsymbol{x})$$

$$\wedge \forall i < y \ DIGIT(c, i+1, M) = h(\boldsymbol{x}, i, DIGIT(c, i, M))$$

$$\wedge DIGIT(c, y, M) = z.$$

Provided that the graphs of g and h are rudimentary, and using the results of the previous section, it is easy to prove that this formula is rudimentary. Of course, this does not mean that the relation $z = f(\boldsymbol{x}, y)$ is rudimentary, because in general the code c is not polynomially bounded in variables $z, y, \boldsymbol{x}$, since we only know $c < M(\boldsymbol{x}, y)^{y+1}$. Nevertheless, in the special case when the function f is increasing enough or small enough, this will be the case, as we shall see in Subsections 3.2 and 3.3.

3.2. Application to Ackermann's functions

Let us precise what we mean by "increasing enough". The main idea is that a fixed number of steps of the calculus can always be processed by hand, as well for the smallest values of the index of recursion as for the biggest ones.

Lemma 3.3. *Let $f(x, \boldsymbol{u})$ be defined from the functions g and h by a primitive recursive schema. If f is increasing relatively to x and verifies*

$$\exists k \ \exists n \ \exists x_0 \ \forall x > x_0 \ \forall \boldsymbol{u} (f(x, \boldsymbol{u}) + 1)^{x+1} < (f(x + k, \boldsymbol{u}))^n$$

and if the graphs of g and h are rudimentary, then the relation $z = f(x, \boldsymbol{u})$ is rudimentary.

Proof. Let us provide the corresponding bounded arithmetical formula (as usual in programming, we insert comments inside it).

$$(x = 0 \wedge f(x, \boldsymbol{u}) = g(\boldsymbol{u})) \vee (x = 1 \wedge f(x, \boldsymbol{u}) = h(x - 1, \boldsymbol{u}, g(\boldsymbol{u})))$$

$$\vee \ldots \vee (x = x_0 + k \wedge f(x, \boldsymbol{u}) = h(x - 1, \boldsymbol{u}, h(x - 2, \boldsymbol{u}, \ldots, h(0, \boldsymbol{u}, g(\boldsymbol{u}))).$$

These two first lines compute the x_0+k+1 first values of f. Note that, since x_0 and k are fixed integers, this disjunction is finite.

$$\vee(x>x_0+k\wedge\exists y_1<y\ldots\exists y_k<y\,\exists c<y^n.$$

Note that, since k is a fixed integer, this formula is still a first-order formula. Moreover, the quantification $\exists c<y^n$ is polynomially bounded because n is also a fixed integer. We are in a position to encode the calculus of $f(0,\boldsymbol{u})$ up to $f(x-k,\boldsymbol{u})$ using the previously defined formula $CODE$ as expected:

$$(CODE^f(y_k,\boldsymbol{u},x-k,c,y_k+1))$$

We achieve the calculus by computing the last k values by hand.

$$\wedge z=h(x,\boldsymbol{u},y_1)\wedge y_1=h(x-1,\boldsymbol{u},y_2)\wedge\ldots\wedge y_{k-1}=h(x-k+1,\boldsymbol{u},y_k))).\quad\square$$

Note that the hypothesis of Lemma 3.3 is very strong. For instance, the function $x\mapsto 2^{2^x}$ is not increasing enough, whereas $x\mapsto \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^x}}}}}}_{\text{height } n_0}$ (for any fixed $n_0\geqslant 4$) works.

Let us apply Lemma 3.3 to Ackermann's partial functions. An alternative (and more complicated) proof of this result can be found in [16].

Definition 3.4. Let us denote by A the following binary function (Ackermann's function)

$$\begin{cases} A(0,y)=y+1\\ A(x+1,0)=A(x,1)\\ A(x+1,y+1)=A(x,A(x+1,y)).\end{cases}$$

The main interest of this function is that it is *not* primitive recursive, because it is increasing too fast (see [22]). Let us now consider the partial functions:

$$\begin{cases} A(0,y)=y+1\\ A(1,y)=y+2\\ A(2,y)=2y+3\\ A(3,y)=2^{y+3}-3\\ A(4,y)=Tower(2,y)-3=\underbrace{2^{2^{\cdot^{\cdot^{\cdot}}}}}_{y\text{ times}}-3.\\ \ldots\end{cases}$$

The following properties are easily verified:

1. $\forall x,y\ A(x,y)<A(x+1,y)$ (the partial functions are bigger and bigger);
2. $\forall x,y\ A(x,y)<A(x,y+1)$ (the partial functions are increasing);
3. $\forall x,y\ A(x,y+1)-A(x,y)<A(x+1,y+1)-A(x+1,y)$ (the partial functions are more and more increasing);
4. $\forall x\geqslant 4,\forall y\,2^{A(x,y)}<A(x,y+1)$ (the partial functions are, except a finite number, at least as increasing as an exponentiation tower);

Lemma 3.5. *The graphs of all partial* Ackermann's *functions are rudimentary.*

Proof. We already know that the graphs of the partial functions $A(0,.)$ up to $A(3,.)$ are rudimentary. Let us show by induction on x that it is true for any $A(x,.)$. Let us fix $x_0>3$ and let us suppose that the binary relation $z=A(x_0-1,y)$ is rudimentary. Then we have:

$$\begin{cases} A(x_0,0)=A(x_0-1,1) \\ A(x_0,y+1)=A(x_0-1,A(x_0,y)) \end{cases}.$$

This is a primitive recursive schema. In order to apply Lemma 3.3, we only have to provide y_0, n and k such that $(\forall y>y_0 (A(x_0,y)+1)^{y+1}<A(x_0,y+k)^n)$. Because of the previous properties, we have $(\forall y\ A(x_0,y+1)^{y+1}<2^{A(x_0,y+1)(y+1)}<2^{A(x_0,y+2)}<A(x_0,y+3))$. Hence $y_0=0$, $n=1$ and $k=3$ fulfill all requirements. □

C. Calude proved in 1987 that *the ternary relation* $z=A(x,y)$ *is rudimentary*, in a technical but likesome way (see [28]).

3.3. *Very short and small recurrences*

There is also another case when the classical encoding $c=\sum_{i=0}^{y} f(\boldsymbol{x},i)M(\boldsymbol{x},y)^i$ is small enough to be used in a bounded arithmetical formula. This case occurs when both the largest value of the index of recursion y and the values of the function are very small relatively to some variable occurring in the parameters $\boldsymbol{x}$.

Lemma 3.6. *Let* $p\geqslant 1$ *be a fixed integer and let* $0<\varepsilon<1$. *Let* ϕ *be any unary function, whose graph lies in* $\mathfrak{R}^{vb}$ *and such that* $\phi(x_1)<\lfloor(\log_2(x_1))^{1-\varepsilon}\rfloor$. *If* (1) $f(\boldsymbol{x},y)$ *is defined by a primitive recursive schema from* g *and* h, (2) *the graphs of* g *and* h *are in* $\mathfrak{R}^{vb}$, (3) *for all* $i\leqslant\phi(x_1)$, *we have* $f(\boldsymbol{x},i)\leqslant(\lfloor\log_2(x_1)\rfloor)^p$, *then the relation* $(y\leqslant\phi(x_1)\wedge(z=f(\boldsymbol{x},y))$ *also lies in* $\mathfrak{R}^{vb}$.

Proof. It suffices to bound the code in a convenient way. We can choose $M(\boldsymbol{x},y)=(\lfloor\log_2(x_1)\rfloor)^p+1$, and the reader easily verifies that the code is $c=\sum_{i=0}^{i=\phi(x_1)} f(\boldsymbol{x},i)M(\boldsymbol{x},y)^i\leqslant x_1$ (for all but a finite number of values of x_1). □

All the functions verifying the hypothesis of Lemma 3.3 are very increasing, and all the functions verifying the hypothesis of Lemma 3.6 are very small. To deal with some other cases, we have to make the code smaller. There are two devices: making the recursion shorter on one hand, and on the other hand making the values of the function smaller. In Section 4, we present two methods of reducing the problem of belonging to $\mathfrak{R}^{vb}$ of the graph of a given primitive recursive function f to that of the graph of another primitive recursive function F, with F satisfying the hypothesis of Lemma 3.6. In Subsection 4.1, we are concerned by making the values of the function smaller, and in Subsection 4.2 with making the recursion shorter.

4. Toolbox

4.1. Chinese remainders coding

Now, we present a tool already used in [19, 26] which allows us to make the values of the function smaller. This coding device is based on theorem 2.4.

First, let us denote *lcm*, the lowest common multiple and let us give two number theoretic lemmas.

Lemma 4.1. *For all* $n \geqslant 1$, *the inequality* $2^{n-1} \leqslant lcm\{k; 1 \leqslant k \leqslant n\}$ *holds.*

Lemma 4.2. *Let* a *and* b *be nonzero integers. If, for all* $k \leqslant \mathrm{MAX}\{\lceil \log(a) \rceil; \lceil \log(b) \rceil\}$, *we have* $rm(a, k+1) = rm(b, k+1)$, *then* $a = b$.

Now, let us come back to recursively defined functions. Let f be defined from g and h by a primitive recursive schema. In order to prove that $z = f(\boldsymbol{x}, y)$, according to Lemma 4.2, it suffices to show that for all $k \leqslant \mathrm{MAX}\{\lceil \log(z) \rceil; \lceil \log(f(\boldsymbol{x}, y)) \rceil\}$, we have $rm(z, k+1) = rm(f(\boldsymbol{x}, y), k+1)$. In case the function $F(\boldsymbol{x}, y, k) = rm(f(\boldsymbol{x}, y), k+1)$ is easier to compute than $f(\boldsymbol{x}, y)$, this will be very useful to our purpose. Let us formalize this idea in Lemma 4.3 below.

Lemma 4.3. *If* (1) f *is defined from* g *and* h *by a primitive recursive schema*; (2) *the graphs of the functions* g *and* h *are in* $\mathfrak{R}^{vb}$; (3) *the function* h *satisfies for all* $y, \boldsymbol{x}, z$ *and* k *the equality*

$$rm(h(\boldsymbol{x}, y, z), k+1) = rm(h(\boldsymbol{x}, y, rm(z, k)), k+1);$$

(4) *the function* f *is bounded by some power of one of the parameters* $\boldsymbol{x}$, *say* x_1, *then for all* $0 < \varepsilon < 1$ *the relation* $(y < \lfloor (\log_2(x_1))^{1-\varepsilon} \rfloor) \wedge (z = f(\boldsymbol{x}, y))$ *is also in* $\mathfrak{R}^{vb}$:

Proof. Let the function G be defined by $G(x_1, k) = \mathrm{Min}\{1 + \lceil \log(x_1) \rceil, k+1\}$. Let us define F by a primitive recursive schema from $rm_{G(x_1,k)}g(\boldsymbol{x})$ and $rm_{G(x_1,k)}(h(\boldsymbol{x}, i, rm(z, k+1)))$. From the hypothesis, for all $k \leqslant \lceil \log(x_1) \rceil$, we have $rm(f(\boldsymbol{x}, y), k+1) = F(\boldsymbol{x}, y, k)$ by an easy induction on y. From Lemma 4.2, we can replace $z = f(\boldsymbol{x}, y)$ by

$$(z \leqslant x_1) \wedge \forall k_{\leqslant \lceil (\log(x_1)) \rceil} (rm(z, k+1) = F(\boldsymbol{x}, y, k)).$$

From the Lemma 3.6, the result is proved. □

At this point, we achieve one of our aims: the values of the function to encode are made much smaller. For instance, if f is bounded by $M(\boldsymbol{x}, y)$, the size of the classical code c of the history of the calculus of $f(\boldsymbol{x}, y)$ is $c < (M(\boldsymbol{x}, y))^{y+1}$, whereas the size of the code C corresponding to F is $C < (\lceil \log_2(M(\boldsymbol{x}, y)) \rceil + 1)^{y+1}$.

4.2. Iteration of recursion

In this subsection, we present a new tool which permits to reduce a recursion of length $y < \lfloor \log_2(x_1) \rfloor^{k+1}$ to at most $2k$ recursions of length $\lfloor \sqrt{\log_2(x_1)} \rfloor$, for any fixed k. The basic idea is quite natural, and comes from parallel computation: to perform a recursion of length $\lfloor \log_2(x_1) \rfloor^{k+1}$, it suffices to perform a recursion of length $\lfloor \sqrt{\log_2(x_1)} \rfloor$, then to iterate it (i.e. perform $\lfloor \sqrt{\log_2(x_1)} \rfloor$ steps of length $\lfloor \sqrt{\log_2(x_1)} \rfloor$), such obtaining a recursion of length $(\lfloor \sqrt{\log_2(x_1)} \rfloor)^2$, and so on, at most $2k$ times (i.e. a fixed number of operations).

Lemma 4.4. *Let P be a polynomial in $\mathbb{N}[X]$ and p be a positive integer. If (1) f is defined from g and h by a primitive recursive schema, (2) the graphs of the functions g and h are in $\mathfrak{R}^{vb}$, (3) the function f is bounded by $\lfloor \log_2(x_1) \rfloor^p$, then the relation $(y < P(\lfloor \log_2(x_1) \rfloor)) \wedge (z = f(\boldsymbol{x}, y))$ is also in $\mathfrak{R}^{vb}$.*

Proof. Let $y < P(\lfloor \sqrt{\log_2(x_1)} \rfloor)$, then for some fixed k, we have $y < (\lfloor \sqrt{\log_2(x_1)} \rfloor)^{2k+1}$. Hence we can write $y = \sum_{j=0}^{2k} a_j (\lfloor \sqrt{\log_2(x_1)} \rfloor)^j$, with $a_j < (\lfloor \sqrt{\log_2(x_1)} \rfloor)$ for $i = 0$ to $2k$. Our goal is to provide $2k+1$ formulas defining the finite sequence $f(\boldsymbol{x}, \sum_{j=0}^{r} a_j (\lfloor \sqrt{\log_2(x_1)} \rfloor)^j)$ for $r = 0$ to $2k$ by induction on r.

First of all, we have to define a new function F_0 corresponding to one use of our transition function h, containing the initial value Z and initial index Y as additional variables. Hence we have:

$$\begin{cases} F_0(\boldsymbol{x}, 0, Z, Y) = \mathrm{Min}\{Z, \lfloor \log_2(x_1) \rfloor^p\} \\ F_0(\boldsymbol{x}, i+1, Z, Y) = \mathrm{Min}\{h(\boldsymbol{x}, Y+i, F_0(\boldsymbol{x}, i, Z, Y)), \lfloor \log_2(x_1) \rfloor^p\}. \end{cases}$$

One can verify by an easy induction that $F_0(\boldsymbol{x}, i, f(\boldsymbol{x}, Y), Y) = f(\boldsymbol{x}, Y+i)$. Now, we define a function denoted by f_0 and corresponding to the recursive application of the transition function h exactly $\lfloor \sqrt{\log_2(x_1)} \rfloor$ times, with initial condition Z and from the index Y, that is $f_0(\boldsymbol{x}, Z, Y) = F_0(\boldsymbol{x}, \lfloor \sqrt{\log_2(x_1)} \rfloor, Z, Y)$. An easy induction shows that F_0 satisfies the hypotheses of Lemma 3.6. Hence, the graph of f_0 is in $\mathfrak{R}^{vb}$.

Then, we iterate this device, and recursively define the functions F_j and f_j corresponding to the recursive use of the function h with steps of length $(\lfloor \sqrt{\log_2(x_1)} \rfloor)^j$, under initial condition Z and from the index Y:

$$\begin{cases} F_{j+1}(\boldsymbol{x}, 0, Z, Y) = \mathrm{Min}\{Z, \lfloor \log_2(x_1) \rfloor^p\} \\ F_{j+1}(\boldsymbol{x}, i+1, Z, Y) = \mathrm{Min}\{f_j(\boldsymbol{x}, F_{j+1}(\boldsymbol{x}, i, Z, Y), Y \\ \qquad\qquad + i(\lfloor \sqrt{\log_2(x_1)} \rfloor)^j), \lfloor \log_2(x_1) \rfloor^p\} \\ f_{j+1}(\boldsymbol{x}, Z, Y) = F_{i+1}(\boldsymbol{x}, \lfloor \sqrt{\log_2(X)} \rfloor, Z, Y). \end{cases}$$

At each step, the graph of the function f_i is in $\mathfrak{R}^{vb}$. One can verify by an easy induction that $F_j(\boldsymbol{x}, i, f(\boldsymbol{x}, Y), Y) = f(\boldsymbol{x}, Y + i(\lfloor \sqrt{\log_2(x_1)} \rfloor)^j)$. Finally, we reach $y = \sum_{j=0}^{2k} a_j (\lfloor \sqrt{\log_2(x_1)} \rfloor)^j$ with $a_j < \lfloor \sqrt{\log_2(x)} \rfloor$ for $i = 0$ up to $2k$ because we compute in

turn $z_0 = F_0(\boldsymbol{x}, a_0, g(\boldsymbol{x}), 0)$ (from the hypothesis, this is in $\mathfrak{R}^{vb}$, because $a_0 < \lfloor\sqrt{\log_2(x_1)}\rfloor$), then $z_1 = F_1(\boldsymbol{x}, a_1, z_0, a_0)$, then $z_2 = F_2(\boldsymbol{x}, a_2, z_0, a_1 \lfloor\sqrt{\log_2(x_1)}\rfloor)$, up to $z = f(\boldsymbol{x}, y) = F_{2k}(\boldsymbol{x}, a_{2k}, z_{2k-1}, \sum_{j=0}^{2k-1} a_j(\lfloor\sqrt{\log_2(x_1)}\rfloor)^j)$. □

As a consequence, using Lemma 4.4 instead of Lemma 3.6 in the proof of Lemma 4.3, we obtain the

Lemma 4.5. *Let P be a polynomial in $\mathbb{N}[X]$. If (1) f is defined from g and h by a primitive recursive schema, (2) the graphs of the functions g and h are in $\mathfrak{R}^{vb}$, (3) the function h satisfies for all $y, \boldsymbol{x}, z, k$, the equality $rm(h(\boldsymbol{x}, y, z), k+1) = rm(h, (\boldsymbol{x}, y, rm(z, k+1)), k+1)$, (4) the function f is bounded by some power of one of the parameters $\boldsymbol{x}$, say x_1, then the relation $(y < P(\lfloor\log_2(x_1)\rfloor)) \wedge (z = f(\boldsymbol{x}, y))$ is also in $\mathfrak{R}^{vb}$.*

5. Application to arithmetical relations

Usually, we need some ad-hoc additional device to reach a recursion of length smaller than $P(\lfloor\log(x_1)\rfloor)$ before applying Lemma 4.5. In this section, we present such devices. We denote the number of element of the set A by $\sharp A$.

5.1. Counting divisors

Definition 5.1. Let us denote by $z = CARDDIV(x)$ the counting binary relation defined by $z = \sharp\{v < x, v | x\}$.

Lemma 5.2. *The relation $z = CARDDIV(x)$ is rudimentary.*

Proof. This result was proved at first by Meloul. Here, we give a new proof.

Let $x = \prod_{i=0}^{s-1} p_i^{\alpha_i}$ be the prime numbers decomposition of x. Hence the number of divisors of x is $\prod_{i=0}^{s-1}(1+\alpha_i)$. Let us define the function f by primitive recursion:

$$\begin{cases} f(x,0) = 1 \\ f(x,i+1) = \begin{cases} f(x,i).(VAL(x, DIVPRIME(x,i)) + 1) & \text{if } i < CARDDIVPRIME(x,i) \\ f(x,i) & \text{otherwise.} \end{cases} \end{cases}$$

The relation $z = CARDDIV(x)$ is then equivalent to $z = f(x, CARDDIVPRIME(x))$. Now, it suffices to verify that the hypothesis of Lemma 4.4 are satisfied:

first, the transition function is defined by

$$\left\{ h(x,i,z) = \begin{cases} z.(VAL(x, DIVPRIME(x,i) + 1) & \text{if } i < CARDDIVPRIME(x,i) \\ z & \text{otherwise.} \end{cases} \right.$$

Consequently it has a rudimentary graph, and obviously satisfies hypothesis (3);

secondly, the function f is less than x. Now, we may apply Lemma 4.5 and, since the function $CARDDIVPRIM(x)$ is less than $\lfloor \log_2(x) \rfloor + 1$, the result is proved. □

5.2. Summing divisors

Lemma 5.3. *The binary relation* $y = \sum_{d|x} d$ *is rudimentary.*

Proof. Let us use the following equalities:

$$\sum_{d|x}^{d} = SUMDIV(x) = \prod_{i=0}^{s-1} \left(\sum_{j=0}^{\alpha_i} p_i^j \right)$$

In order to give rudimentary definitions of each of the unbounded sum involved, let us consider the recursively defined function $f(x,i,j)$:

$$\begin{cases} f(x,i,0) = 1 \\ f(x,i,j+1) = \begin{cases} f(x,i,j) + DIVPRIME(x,i)^{j+1} \\ \qquad\qquad \text{if } j < VAL(x, DIVPRIME(x,i)) \\ f(x,i,j) \qquad \text{otherwise.} \end{cases} \end{cases}$$

We have $f(x,i,VAL(x,DIVPRIME(x,i))) = \sum_{j=0}^{\alpha_i} p_i^j$ and the function f satisfies all the hypotheses of Lemma 4.5. In order to give rudimentary definition of the product, we also define the function $g(x,i)$ as follows:

$$\begin{cases} g(x,0) = 1 \\ g(x,i+1) = \begin{cases} g(x,i).f(x,i,VAL(x,DIVPRIME(x,i))) \\ \qquad\qquad \text{if } i < CARDDIVPRIME(x) \\ g(x,i) \qquad \text{otherwise.} \end{cases} \end{cases}$$

We have $g(x, CARDDIVPRIME(x)) = SUMDIV(x)$ and the function g satisfies all the hypotheses of Lemma 4.5. Hence, the relation $z = SUMDIV(x)$ is rudimentary. □

Definition 5.4. An integer x is *perfect* iff it is equal to the sum of its strict divisors.

Corollary 5.5. *The set of perfect numbers is rudimentary.*

Proof. Obvious since x is a perfect number iff $2x = \sum_{d|x} d$. □

6. Application to linear and polynomial recurrences

In this section, we generalize the proof of the rudimentarity of the sequence of Fibonacci's numbers (see [6, 16]) to the case of linear and polynomial recurrences.

Definition 6.1. A sequence of natural integers $(u_n)_{n\geqslant 0}$ is a linear recurrence if there exist $k>0$ and integers $a_1,\ldots,a_k$ such that for all $n\geqslant k$, $u_n=\sum_{i=1}^{k} a_i u_{n-i}$.

The only non-trivial case occurs when $2\leqslant \sum_{i=1}^{k} a_i$ and $(u_0,\ldots,u_{k-1})\neq(0,\ldots,0)$. Now, let us state two technical results that provide an ad hoc device for making the recursive calculus of u_n shorter and smaller.

Lemma 6.2. *Let $(u_n)_{n\geqslant 0}$ be a linear recurrence such that $2\leqslant \sum_{i=1}^{k} a_i$ and $(u_0,\ldots,u_{k-1})\neq(0,\ldots,0)$. Then there exist integers n_0, $A\geqslant 2$, $s\geqslant 1$ and $B\geqslant 2$ such that for all $n>n_0$, we have $A^n<(u_n)^s$ and $u_n<B^n$.*

Proof. By induction on n. □

Before applying our tools, we have to shorten the recursion. To this end, we use the periodicity of the sequence of the remainders: for any $c>0$, let us denote by $(u_n^c)_{n\geqslant 0}$ the sequence of remainders modulo $c+1$ of the linear recurrence $(u_n)_{n\geqslant 0}$, i.e. $u_n^c=rm(u_n,c+1)$.

Lemma 6.3. *There exist integers T (called period) and d (called the beginning of periodicity) such that $d+T\leqslant(1+c)^k$ and for all n greatest than d, we have $u_{n+T}^c=u_n^c$.*

Proof. Consider the set $X=\{(u_n^c,u_{n+1}^c,\ldots,u_{n+k-1}^c),\ 0\leqslant n\leqslant(1+c)^k\}$. Due to the inequality $c\geqslant u_n^c$ for every n, the cardinality of X is at most $(1+c)^k$. Thanks to the restriction of the domain of n, there are in fact $(1+c)^k+1$ such k-tuples. Hence at least two k-tuples are equal. This fact, together with the definition of $(u_n^c)_{n\geqslant 0}$, proves the lemma. □

The definition of the sequence $(u_n^c)_{n\geqslant 0}$ is *not* a primitive recursive schema. Now, our task consists in reducing the problem to some primitive recursive schema. Let us present a classical coding device (see [22]). The idea consists in replacing the sequence of integers $(u_n^c)_{n\geqslant 0}$ by the sequence $(U_n^c)_{n\geqslant 0}$ of elements of $\mathbb{N}^k$ defined by $U_n^c=(u_n^c,\ldots,u_{n-k+1}^c)$. We encode the k-tuple U_n^c into one single integer. We choose a coding function for k-tuples which is a polynomial of degree k; we denote such a polynomial by $POL(x_1,\ldots,x_k)$. We do not need to explicit this polynomial. We only need the existence of decoding functions, denoted by $UNC_i(y)$ for $i=1$ to n. These functions satisfy $UNC_i(POL(x_1,\ldots,x_k))=x_i$. It can be shown that the graphs of these decoding functions are rudimentary (see [1]).

Now, using the transition function

$$h(z)=POL\left(rm_{\mathrm{mod}(c)}\left(\sum_{i=1}^{k} a_i UNC_i(z)\right),UNC_1(z),\ldots,UNC_{k-1}(z)\right),$$

we define the coding sequence $(v_n^c)_{n\geqslant 0}$ for $(U_n^c)_{n\geqslant 0}$ by

$$\begin{cases} v_0^c = POL(u_{k-1}^c,\ldots,u_0^c)\\ v_{n+1}^c = h(v_n^c). \end{cases}$$

This is a primitive recursive schema defining the sequence $(v_n^c)_{n\geqslant 0}$, showing as well its study is in the scope of this paper.

Theorem 6.4. *All linear recurrences are rudimentary relations.*

Proof. We first show that the sequence $(u_n)_{n\geqslant 0}$ is rudimentary if and only if for all $c\leqslant \lceil \log_2(B^n)\rceil$ the sequence $(u_n^c)_{0\leqslant n\leqslant (c+1)^k+1}$ is rudimentary. Then we prove this fact, using the encoded sequence $(v_n^c)_{n\geqslant 0}$ and a legal additional parameter z such that $A^n<z^s$ and $z<B^n$.

Let p be an integer such that $B<A^p$. From Lemmas 4.2 and 6.2, the relation $z=u_n$ is equivalent to

$$(z=u_0\wedge n=0)\vee\ldots\vee(z=u_{n_0}\wedge n=n_0)$$
$$\vee(n>n_0\wedge z<B^n<z^{sp}\wedge(\forall c)\leqslant\lceil\log_2(B^n)\rceil rm(z,c+1)=rm(u_n,c+1)).$$

Therefore, our problem is reduced to that of the rudimentarity of the sequence $(u_n^c)_{n\geqslant n_0}$. Up to a translation of the index of recursion, we can assume that $n_0=0$. Now, Lemma 6.3 provides the following formula denoted by $DEF(d,T,c)$ defining a period T and a beginning of periodicity d for $(u_n^c)_{n\geqslant n_0}$:

$$d\leqslant(c+1)^k\wedge\exists n\leqslant(c+1)^k(n>d\wedge u_n^c=u_d^c\wedge T=n-d).$$

Consequently, we have $u=u_n^c$ iff

$$\forall T\leqslant(c+1)^k\forall d\leqslant(c+1)^k[DEF(d,T,c)\rightarrow([n\leqslant d\wedge u=u_n^c]$$
$$\vee[n>d\wedge\exists r\leqslant(c+1)^k+1(d+T>r\geqslant d)\wedge(r\equiv n(\mathrm{mod}\,T)\wedge u=u_r^c)])].$$

Hence what we actually need are the $(c+1)^k+2$ first values of the sequence $(u_n^c)_{n\geqslant 0}$, as we announced before. Since $c\leqslant\lceil\log_2(B^n)\rceil$, we have $u_n^c<1+2sp\lfloor\log_2(z)\rfloor$ (hence $v_n^c<(\lfloor\log_2(z)\rfloor)^q$ for some q) and $(c+1)^k+2<(2+2sp\lfloor\log_2(z)\rfloor)^k+2$, so that the hypotheses of Lemma 4.4 are satisfied for the sequence $(v_n^c)_{0\leqslant n\leqslant(c+1)^k+1}$. This completes the proof. □

Now, let us pass to the case of polynomial recurrences. It is obtained from the previous case of linear recurrences by minor changes in the proofs.

Definition 6.5. A sequence of natural integers $(u_n)_{n\geqslant 0}$ is a polynomial recurrence if there exist $k>0$ and a polynomial $P(x_1,\ldots,x_k)$ such that $\forall n\geqslant k$, $u_n=P(u_{n-1},\ldots,u_{n-k})$.

Theorem 6.6. *All polynomial recurrences are rudimentary relations.*

7. On polynomial substitution in counting rudimentary sets

In this section, we use the tools we present in this paper from a slightly different viewpoint. Indeed, we are no more concerned in proving that a given primitive recursive

relation is rudimentary. Our aim is to prove that counting rudimentary relations are closed under polynomial substitution as defined in 7.3 below.

Definition 7.1. Let $R(x_1,\ldots,x_n)\subset\mathbb{N}^n$. We say that the relation $S(x_1,\ldots,x_n,z)$ is obtained from R by a *counting operation* when for some index $l\in\{1,\ldots,n\}$ we have $S(x_1,\ldots,x_n,z)$ iff $z=\sharp\{i<x_l/R(x_1,\ldots,x_{l-1},i,x_{l+1},\ldots,x_n)\}$.

Definition 7.2. Let us denote by $\mathfrak{R}^\sharp$ the smallest class of relations over integers containing the graph of addition and multiplication and closed under the following operations:

- boolean operations; • explicit transformations; • counting operation; • bounded quantifications $\forall x<y\ldots$ and $\exists x<y\ldots$.

It is easy to verify that a counting operation is a very restrictive primitive recursive schema. Hence, the class $\mathfrak{R}^\sharp$ is an $\mathfrak{R}^{vb}$ class, and we can use Lemmas 4.3 to 4.5 in this framework. Moreover, since $\forall i<y\,R(\boldsymbol{x},i)$ is equivalent to $y=\sharp\{i<y/R(\boldsymbol{x},i)\}$, we can drop the requirement on quantifications in this definition without loss of generality.

The best-known counting rudimentary relation is $y=p_x$ meaning that y is the xth prime number (i.e. $p_0=2$, $p_1=3$, $p_2=5,\ldots$). It *is not known whether* $\mathfrak{R}^\sharp=\mathfrak{R}$. For instance, *the question of whether the relation* $y=p_x$ *is in* $\mathfrak{R}$ *is still open.* In [9], an arithmetical definition of this relation, in which all quantifications are exponentially bounded is provided. In [4, 16], the authors provide rudimentary definitions of such relations as $y=(p_x)^{\lfloor\sqrt{x}\rfloor}$ and $y=(p_x)^x$.

The class $\mathfrak{R}^\sharp$ was first investigated by Paris and Wilkie in [19]. They proved that, in the special case when $R(x_1,\ldots,x_n)$ is a rudimentary relation and the number z of $i<y$ such that $R(x_1,\ldots,x_{l-1},i,x_{l+1},\ldots,x_k)$ verifies $z<\lfloor\log_2(x_l)\rfloor^n$ (for some fixed n and l), then the relation $z=\sharp\{i<y/R(x_1,\ldots,x_{l-1},i,x_{l+1},\ldots,x_k)\}$ is also rudimentary.

Definition 7.3. Let R be a relation over n, of arity k. We say that the relation $S(\boldsymbol{x})$ is obtained from R by a polynomial substitution if for some k-tuple of polynomials $P_1(\boldsymbol{x}),\ldots,P_k(\boldsymbol{x})$ in $\mathbb{N}[X_1,\ldots,X_k]$, we have $S(\boldsymbol{x})$ iff $R(P_1(\boldsymbol{x}),\ldots,P_k(\boldsymbol{x}))$.

Let us recall that the following inclusions hold: $\mathfrak{R}\subseteq\mathfrak{R}^\sharp\subseteq\mathfrak{I}_*^{-1}\subseteq\mathfrak{E}_*^0\subseteq\mathfrak{E}_*^2\subset\mathfrak{E}_*^3$ and that only the last one is known to be strict. Also, it is known that the closure under polynomial substitution of $\mathfrak{E}_*^0$ and of $\mathfrak{I}_*^{-1}$ are $\mathfrak{E}_*^2$ (see [4]), whereas $\mathfrak{R}$ and $\mathfrak{E}_*^2$ are closed under polynomial substitution (see [9]). Here we prove that it is also the case for $\mathfrak{R}^\sharp$. Moreover, it is known that $\mathfrak{E}_*^3$ (so-called *arithmetical relations*) is closed under exponential substitution, whereas the closure under exponential substitution of $\mathfrak{R}$ (hence also of $\mathfrak{R}^\sharp,\mathfrak{I}_*^{-1},\mathfrak{E}_*^0$, and $\mathfrak{E}_*^2$) is $\mathfrak{E}_*^3$.

Theorem 7.4. *The class* $\mathfrak{R}^\sharp$ *is closed under polynomial substitution.*

Before proving it, let us notice that Theorem 7.4 is equivalent to Corollary 7.5 below.

Corollary 7.5. *The class* $\mathfrak{R}^{\#}$ *is closed under polynomially bounded quantifications.*

The following corollary emphasizes the important role of Theorem 7.4 in these topics.

Corollary 7.6. *If* $\mathfrak{R}^{\#} = \mathfrak{I}_*^{-1}$, *then* $\mathfrak{R}^{\#} = \mathfrak{I}_*^{-1} = \mathfrak{E}_*^0 = \mathfrak{E}_*^1 = \mathfrak{E}_*^2$.

Proof. It follows from the fact that $\mathfrak{E}_*^2$ is the closure of $\mathfrak{I}_*^{-1}$ by polynomial substitutions. (see [4]). □

Now, let us turn back to the proof of Theorem 7.4. We have divided it into three steps. We first show that Theorem 7.4 is a consequence of Lemma 7.8. Then we show that Lemma 7.8 is a consequence of Lemma 7.9. Finally, we prove Lemma 7.9. For simplicity of notation, let us introduce the following definition.

Definition 7.7. Let R be a relation in $\mathfrak{R}^{\#}$ of arity k. We say that R satisfies property *POLSUBST* if for any k-tuple of polynomials $P_1(\boldsymbol{x}),\ldots,P_k(\boldsymbol{x})$ in $\mathbb{N}[x_1,\ldots,x_n]$, the relation $R(P_1(\boldsymbol{x}),\ldots,P_k(\boldsymbol{x}))$ still lies in $\mathfrak{R}^{\#}$.

Hence, Theorem 7.4 can be rephrased as: Any relation $R \in \mathfrak{R}^{\#}$ satisfies *POLSUBST*. Below, every polynomial will be in $\mathbb{N}[x_1,\ldots,x_n]$.

First step

Lemma 7.8. *For all relation* $R(\boldsymbol{x}, y) \in \mathfrak{R}^{\#}$ *satisfying POLSUBST, the relation defined by* $z = \#\{i < y / R(\boldsymbol{x}, i)\}$ *also satisfies POLSUBST.*

We intend to prove that the Theorem 7.4 is a consequence of Lemma 7.8. by external induction on the construction of the relations of $\mathfrak{R}^{\#}$.

1. The relations $x_1 = x_2 + x_3$ and $x_1 = x_2 . x_3$ satisfy *POLSUBST* because any polynomial relation lies in $\mathfrak{R}$, hence also in $\mathfrak{R}^{\#}$.
2. If R and Q both satisfy *POLSUBST*, it is easy to verify that $\neg R, R \wedge Q$ and $R \vee Q$ also satisfy it.
3. If $R(\boldsymbol{x}, y) \in \mathfrak{R}^{\#}$ satisfies *POLSUBST*, then Lemma 7.8 ensures that $z = \#\{i < y / R(\boldsymbol{x}, i)\}$ also satisfies *POLSUBST*.

Second step

Let us turn to the proof of Lemma 7.8. We proceed by simultaneous substitutions of polynomials to all variables. With this aim in view, we introduce Lemma 7.9. We shall denote $x'_1,\ldots,x'_m$ by $\boldsymbol{x}'$ and $\pi_1(\boldsymbol{x}),\ldots,\pi_m(\boldsymbol{x})$ by $\boldsymbol{\pi}(\boldsymbol{x})$ and for any fixed integer p, let $D_p(d,x)$ be the integer $\inf\{d+1, 1+\lceil p \log_2(2+x)\rceil\}$.

Lemma 7.9. *Let $R(y,\boldsymbol{x}')$ be a relation in $\mathfrak{R}^{\#}$ satisfying POLSUBST; for any polynomial $P(\boldsymbol{x})$, for any $(m+1)$-tuple of polynomials $(\pi_0(\boldsymbol{x},u),\boldsymbol{\pi}(\boldsymbol{x}))$, and any index $1\leqslant l\leqslant n$, and any integer p, the graph of the function F defined by*

$$F(d,\boldsymbol{x})=rm_{\mathrm{mod}(D_p(d,x_l))}\#\{i\leqslant P(\boldsymbol{x});R(\pi_0(\boldsymbol{x},i),\boldsymbol{\pi}(\boldsymbol{x}))\}$$

lies in $\mathfrak{R}^{\#}$.

We intend to prove that Lemma 7.8 is a consequence of Lemma 7.9. Let $R(y,\boldsymbol{x}')$ be a relation in $\mathfrak{R}^{\#}$ satisfying *POLSUBST*. In order to prove that the relation $z=\#\{i<y/R(i,\boldsymbol{x}')\}$ also satisfies *POLSUBST*, let $(P(\boldsymbol{x}),\boldsymbol{\pi}(\boldsymbol{x}),\psi(\boldsymbol{x}))$ be a $(m+2)$-tuple of polynomials. Let us choose an integer p such that $P(\boldsymbol{x})\leqslant(2+\mathrm{Max}\{x_l;1\leqslant l\leqslant n\})^p$. Then the relation $\psi(\boldsymbol{x})=\#\{i<P(\boldsymbol{x})/R(i,\boldsymbol{\pi}(\boldsymbol{x}))\}$ is equivalent to

$$(\psi(\boldsymbol{x})\leqslant P(\boldsymbol{x}))\wedge\bigvee_{l=1}^{l=n}\left(\bigwedge_{\lambda=1}^{\lambda=n}\right)(x_\lambda\leqslant x_l)\wedge(\forall d)_{\leqslant\lceil p\cdot\log_2(2+x_l)\rceil}(\exists z)_{\leqslant d}$$
$$(\psi(\boldsymbol{x})\equiv z(\mathrm{mod}(d+1)))\wedge(z=rm_{\mathrm{mod}\,D_p(d,x_l)}\#\{i<P(\boldsymbol{x});R(i,\boldsymbol{\pi}(\boldsymbol{x}))\})).$$

It follows from Lemma 7.9 that this relation lies in $\mathfrak{R}^{\#}$.

Last step

Now, let us prove Lemma 7.9. The proof proceeds by induction on the construction of the polynomial P. Let $R(y,\boldsymbol{x}')$ be a relation in $\mathfrak{R}^{\#}$ satisfying *POLSUBST*.

Case of a sum of polynomials. Let $P_1(\boldsymbol{x})$ and $P_2(\boldsymbol{x})$ be such that for any $(m+1)$-tuple of polynomials $(\pi_0(\boldsymbol{x},u),\boldsymbol{\pi}(\boldsymbol{x}))$, and for any index $1\leqslant l\leqslant n$, and for any integer p, the graph of the two functions F_1 and F_2 defined by $F_1(d,\boldsymbol{x})=rm_{\mathrm{mod}\,D_p(d,x_l)}\#\{i\leqslant P_1(\boldsymbol{x});R(\pi_0(\boldsymbol{x},i),\boldsymbol{\pi}(\boldsymbol{x}))\}$ and $F_2(d,\boldsymbol{x})=rm_{\mathrm{mod}\,D_p(d,x_l)}\#\{i\leqslant P_2(\boldsymbol{x});R(\pi_0(\boldsymbol{x},i+P_1(\boldsymbol{x})),\boldsymbol{\pi}(\boldsymbol{x}))\}$ are in $\mathfrak{R}^{\#}$.

Then the relation $rm_{\mathrm{mod}\,D_p(d,x_l)}\#\{i\leqslant P_1(\boldsymbol{x})+P_2(\boldsymbol{x});R(\pi_0(\boldsymbol{x},i),\boldsymbol{\pi}(\boldsymbol{x}))\}=z$ is equivalent to

$$(\exists z_1)_{<D_p(d,x_l)}(\exists z_2)_{<D_p(d,x_l)}(z_1=F_1(d,\boldsymbol{x}))\wedge(z_2=F_2(d,\boldsymbol{x}))$$
$$\wedge(\neg R(\pi_0(\boldsymbol{x},P_1(\boldsymbol{x})),\boldsymbol{\pi}(\boldsymbol{x}))\wedge(z=rm_{\mathrm{mod}(D_p(d,x_l)}(z_1+z_2))$$
$$\vee R(\pi_0(\boldsymbol{x},P_1(\boldsymbol{x})),\boldsymbol{\pi}(\boldsymbol{x}))\wedge(z=rm_{\mathrm{mod}(D_p(d,x_l)}(z_1+z_2-1)))$$

and consequently lies in $\mathfrak{R}^{vb}$.

Case of a product by a variable, say x_1. Let $P(\boldsymbol{x})$ be a polynomial such that the graph of the function $F(d,\boldsymbol{x})=rm_{\mathrm{mod}\,D_p(d,x_l)}\#\{i<P(\boldsymbol{x})/R(\pi_0(\boldsymbol{x},i)),\boldsymbol{\pi}(\boldsymbol{x})\}$ is in $\mathfrak{R}^{\#}$ for any $\boldsymbol{\pi}(\boldsymbol{x}),\pi_0(\boldsymbol{x},v),l$ and p as above. Let us consider the function $F'(d,\boldsymbol{x})=rm_{\mathrm{mod}\,D_p(d,x_l)}$ $\#\{i<x_1\cdot P(\boldsymbol{x})/R(\pi_0(\boldsymbol{x},i)),\boldsymbol{\pi}(\boldsymbol{x}),)\}$. Let $G_p(\boldsymbol{x},d)$ be defined by

$$\sum_{j=0}^{D_p(d,x_l)-1}rm_{\mathrm{mod}\,D_p(d,x_l)}j.\#\{u<x_1;\#\{v<P(\boldsymbol{x});R(uP(\boldsymbol{x})+v,\boldsymbol{\pi}(\boldsymbol{x})\}$$
$$\equiv j(\mathrm{mod}(D_p(d,x_l)))\}.$$

Roughly speaking, this function is obtained at first by counting of the intervals of length $P(\boldsymbol{x})$ inside of which the number of integers such that $R(\pi_0(\boldsymbol{x}, i), \pi(\boldsymbol{x}))$ is equal to j modulo $D(d, x_l)$; secondly, by multiplying this number by j; at last by summing modulo $D(d, x_l)$ all these numbers up to $D(d, x_l) - 1$.

Now, using the induction hypothesis on P, the relation $\sharp\{v < P(\boldsymbol{x}); R(uP(\boldsymbol{x}) + v, \pi(\boldsymbol{x})\} \equiv j(\mathrm{mod}(D_p(d, x_l)))$ is in $\mathfrak{R}^\sharp$. Then, the counting relation $y = \sharp\{u < x_1 / \sharp\{v < P(\boldsymbol{x}) / R(u.P(\boldsymbol{x}) + v, \pi(\boldsymbol{x}))\} \equiv j(\mathrm{mod}(D_p(d, x_l)))\}$ is also in $\mathfrak{R}^\sharp$. Finally, using Lemma 4.4, we show that the graph of G_p is in $\mathfrak{R}^\sharp$. Thanks to the equality between $G_p(\boldsymbol{x}, d)$ and $F'(d, \boldsymbol{x})$ the proof is complete. □

References

[1] J.H. Bennett, On Spectra, Doctoral Dissertation, Princeton University, Princeton, NJ, 1962.

[2] P. Cegielski, Le théorème de Dirichlet est finitiste, Preprint LITP 92.40, Université Paris 6-Paris 7, 1992.

[3] C. Dimitracopoulos, Matijasevic's theorem and fragments of arithmetic, Ph. D. Thesis, University of Manchester, 1980.

[4] H.A. Esbelin, Relations rudimentaires et petites classes de fonctions récursives. Doctoral Dissertation, Université d'Auvergne, 1994.

[5] H.A. Esbelin, Une classe minimale de fonctions récursives contenant les relations rudimentaires, C. R. Acad. Sci. Paris, Série I, Paris, 1994, pp. 505–508.

[6] H.A. Esbelin, M. More, L'ensemble des nombres de Fibonacci est rudimentaire, Preprint LLAIC1, 39, Université d'Auvergne, 1994.

[7] K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. Monatshefte für Math. Phys. 38 (1931) 173–198.

[8] A. Grzegorczyk, Some Classes of Recursive Functions, Rozprawy Matematyczne 4 (1953) 1–46.

[9] K. Harrow, Sub-elementary classes of functions and relations, Doctoral Dissertation, New York University, Department of Mathematics, 1973.

[10] K. Harrow, Small Grzegorczyk classes and limited minimum, Z. Math. Logik Grundlagen Math. 11 (1975) 417–426.

[11] N.D. Jones, Context-free languages and rudimentary attributes, Math. System Theory 3 (1969) 102–109.

[12] R.M. Karp, Reducibility among combinatorial problems, IBM Symp. 1972, Complexity of Computers Computations, Plenum Press, New York, 1972.

[13] M. Kutylovski, Small Grzegorczyk classes, J. London Math. Soc. 2 (36) (1987) 193–210.

[14] J. Malifaud, Contribution à l'étude des fonctions calculables de croissance limitéé, Doctoral Dissertation, Université Paris 7, 1985.

[15] J. Meloul, Rudimentary predicates, low complexity classes and related automata, Ph.D. Dissertation, Oxford University, 1979.

[16] M. More, Rudimentary representations of spectra, Prépublication du LLAICI, Numéro 40, 1994.

[17] M. More, F. Olive, Rudimentary languages and second-order logic, Préprint GRAL, 1996-1, 1996.

[18] G. Myhill, Linear bounded automata, Wright Air Devel. Div., Tech. Note 60–165, Wright-Patterson Air Force Base, OH, 1960.

[19] J.B. Paris, A.J. Wilkie, Counting Δ_0 sets, Fund. Math. 127 (1987).

[20] W. Quine, Concatenation as a basis for arithmetic from a logical point of view, J. Symbol. Logic 11 (1946) 105–114. Selected Logic Papers, Random House, 1966, ch. V, pp. 70–82.

[21] R.W. Ritchie, Classes of Predictably Computable Functions, Trans. Amer. Math. Soc. 106 (1963) 139–173.

[22] H.Jr. Rogers, Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.

[23] H. Shapiro, Introduction to the Theory of numbers, Wiley, New York, 1982.

[24] R. Smullyan, Theory of Formal Systems, Ann. Math. Studies, vol. 47, Princeton University Press, Princeton, NJ, 1961.
[25] L.J. Stockmeyer, The polynomial time hierarchy, Theor. Comput. Sci. 3 (1976) 1–22.
[26] A. Woods, Some problems in logic and number theory, and their connections, Doctoral Dissertation, University of Manchester, 1981.
[27] C. Wrathall, Rudimentary predicates and relative computation, SIAM J Comput. 7(2) (1978) 194–209.
[28] C. Calude, Super-exponential non primitive recursive, but rudimentary, Information Processing Letters 25 (1987) 311–315.