Theoretical Computer Science 257 (2001) 51–77

Theoretical Computer Science

www.elsevier.com/locate/tcs

Decidability of the theory of the natural integers with the cantor pairing function and the successor

Patrick Cegielski[a, *], Denis Richard[b]

[a]*LACL, Université Paris XII, Route forestière Hurtault, 77300 Fontainebleau, France*
[b]*LLAIC1, Université d'Auvergne, I.U.T. Informatique des Cézeaux, B.P. 86, F-63172 Aubiere, Cedex, France*

Abstract

The binary Cantor pairing function C from $\mathbb{N} \times \mathbb{N}$ into $\mathbb{N}$ is defined by $C(x, y) = (\frac{1}{2})(x + y)(x + y + 1) + y$. We consider the theory of natural integers equipped with the Cantor pairing function and an extra relation or function X on $\mathbb{N}$. When X is equal either to multiplication, or coprimeness, or divisibility, or addition or natural ordering, it can be proved that the theory $Th(\mathbb{N}, C, X)$ is undecidable. Let S be the successor function. We provide an algorithm solving the decision problem for $Th(\mathbb{N}, C, S)$. © 2001 Elsevier Science B.V. All rights reserved.

0. Introduction

In 1873 [1, 2], Cantor has proved sets $\mathbb{N}^2$ and $\mathbb{N}$ are equipotent, where $\mathbb{N}$ is the set of natural numbers, exhibiting a one-to-one mapping defined by

$$C(x, y) = (1/2)(x + y)(x + y + 1) + y.$$

There exist many other one-to-one mappings and injections from $\mathbb{N}^2$ to $\mathbb{N}$. A *pairing function* J is an injection from $\mathbb{N} \times \mathbb{N}$ in $\mathbb{N}$. It is interesting, for many reasons, to study the elementary theory of such structures $(\mathbb{N}, J)$. We know [4] that there exists pairing function J_0 such that $(\mathbb{N}, J_0)$ is decidable and (recursive) pairing function J_1 such that $(\mathbb{N}, J_1)$ is undecidable. No result existed on well-known pairing functions. A lower bound is known on computational complexity of structures $(\mathbb{N}, J)$ [10, 7, Chapter 8, 5]: *If f is a one-to-one function from A^2 in A the computational complexity of the first-order theory of the structure* $(A, f, =)$ has a lower bound in

$NTime(\exp_\infty(O(n)))$, where $\exp_\infty(n)$is an exponetial tower $2^{2^{\cdot^{\cdot^{2}}}}$ of height n.

[*] Corresponding author.
E-mail addresses: cegielski@univ-paris12.fr (P. Cegielski), richard@llaic.u-clermont1.fr (D. Richard).

0304-3975/01/$ - see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0304-3975(00)00109-2

We are interested in the most famous of these pairing functions: the *Cantor pairing function C* is defined above. A permutation between x and y provides the symmetrical Cantor pairing function whose investigation is obviously the same to the previous. The theory $Th(\mathbb{N}, C)$ is decidable as shown by Cegielski et al. [11]. The proof of this result is based on Malcev theorem on decidability of free semi-groups without relations; $C(0,0)=0$ and some other relations need some special arguments to permit a reduction to the situation solved by Malcev. Our concern is the theory $TH(\mathbb{N}, C, R)$ where R is one of the following predicates or functions: multiplication, divisibility, addition, order or successor. It turns out the case of the last one is the unique to be decidable among considered predicates and functions. The present paper is only devoted to prove this decidability result.

1. Expansion of the language

To prove $TH(\mathbb{N}, C, R)$ is decidable, we eliminate quantifiers in an effective manner within an expansion by definition.

1.1. *Definition of expanded languages*

In a short first step, we investigate the general structure $(\mathbb{N}, J, S)$, for any pairing function J. Let us notice the right and left inverse maps we denote, following Julia Robinson [9], by K and L, are definable in the structure $(\mathbb{N}, J)$ since we have

$$x = K(y) \leftrightarrow \exists u J(x,u) = y,$$

$$x = L(y) \leftrightarrow \exists u J(u,x) = y.$$

The constant 0 is also definable in the structure $(\mathbb{N}, S)$:

$$x = 0 \leftrightarrow \forall y (Sy \neq x).$$

The predecessor function P is similarly defined by $P(x+1)=x$ and $P(0)=0$.

We shall consider the languages $\mathscr{L}=\{J,K,L,S\}$, $\mathscr{M}=\{K,L,S,P,0\}$, and $\mathscr{N}=\{K,L\}$. In compliance with the syntax of first-order languages, $\mathscr{L}$-terms, $\mathscr{M}$-terms, $\mathscr{N}$-terms, atomic $\mathscr{L}$-formulas and $\mathscr{M}$-formulas are presented by structural induction.

Definition 1.1.1. The set of *$\mathscr{L}$-terms* is defined as follows:
(i) any variable is an $\mathscr{L}$-term;
(ii) if t is an $\mathscr{L}$-term then $K(t)$, $L(t)$ and $S(t)$ are also $\mathscr{L}$-terms;
(iii) if t and u are *$\mathscr{L}$-terms* then $J(t,u)$ is also a $\mathscr{L}$-term.

Definition 1.1.2. The set of *$\mathscr{M}$-terms* is defined as follows:
(i) 0 is an $\mathscr{M}$-term;
(ii) any variable is an $\mathscr{M}$-term;
(iii) if t is an $\mathscr{M}$-term then $K(t)$, $L(t)$, $S(t)$, and $P(t)$ are also $\mathscr{M}$-terms.

Definition 1.1.3. The set of $\mathscr{N}$-terms is defined as follows:
(i) any variable is an $\mathscr{N}$-term;
(ii) if t is an $\mathscr{N}$-term then $K(t)$ and $L(t)$ are $\mathscr{N}$-terms.

We must note that in an $\mathscr{L}$-term, as many variables as we want may occur but only one variable, say x, occurs in an $\mathscr{N}$-term t so that in this case we denote t by $t(x)$. As usual, the term we obtain by substitution of u to the variable x is written as $t(u)$. In an $\mathscr{M}$-term at the most one variable occurs.

Definition 1.1.4. (1) An *atomic $\mathscr{L}$-formula* is of the form $t(\bar{x}) = u(\bar{y})$, where t and u are $\mathscr{L}$-terms and $\bar{x}$ and $\bar{y}$ are finite sequences of variables.

(2) An *atomic $\mathscr{M}$-formula* is of the form $t(x) = u(y)$, where t and u are $\mathscr{M}$-terms and x and y are variables or the constant 0.

(3) An *atomic $\mathscr{N}$-formula* is of the form $t(x) = u(y)$, where t and u are $\mathscr{N}$-terms and x and y are variables.

1.2. Elimination of the symbol J

Let us begin by eliminating from $\mathscr{L}$-formulas all occurrences of the functional symbol J. In order to do this, we show in next lemma that any $\mathscr{L}$-formula is equivalent in the structure $\langle \mathbb{N}, J, K, L, S, P, O \rangle$ to an $\mathscr{M}$-formula lemma.

Lemma 1.2.1. *Any $\mathscr{L}$-formula is equivalent to an $\mathscr{M}$-formula.*

Proof. This comes from the fact J is $\mathscr{M}$-definable by $J(x, y) = z \leftrightarrow [K(z) = x \wedge L(z) = y]$. $\square$

1.3. Normalization of terms

From now and then we consider the special case when J is the Cantor pairing function C. A non-closed $\mathscr{M}$-term is characterized by its variable and by a finite sequence of occurrences of the functions K, L, S and P. We prove that one can normalize these terms by putting all S and P at the head of the sequence. Moreover, we can insure any $\mathscr{M}$-term $t(x)$ is equivalent to a normal form according to some condition on x expressed by a $\mathscr{N}$-term. Although, natural order on integers is not easily definable in our languages $\mathscr{L}$, $\mathscr{M}$ and $\mathscr{N}$ and in order to simplify notations, we may make a free use of $x \geqslant n$ for a fixed integer constant n.

Definition 1.3.1. As usual, the term $SSS\ldots Sx$ (resp. $PPP\ldots Px$) with n occurrences of the symbol S (resp. P) is denoted by $x + n$ (resp. $x - n$). We denote by $x \geqslant n + 1$ the conjunction $x \neq 0$, $x \neq 1, \ldots, x \neq n$.

Proposition 1.3.1 (Externalization of the symbols S and P). *For any $\mathscr{M}$-term $t(x)$, where x is a variable, there exist an integer m and $m + 1$ conditions $H_i(x)$ of one of* *the forms* $u_0(x)=0$, *or* $u_1(x)\neq 0$, *or* $\dots$, *or* $u_{m-1}(x)=m-1$, *or* $u_m(x)\geqslant m$, *where* u_i *is an* $\mathscr{N}$*-term and* $m+1\mathscr{N}$*-terms* $v_i(x)$ *and the integers* k_i*'s are positive, negative or null integers such that, if* $H_i(x)$ *holds then* $t(x)=v_i(x)+k_i$.

Such a condition $H_i(x)$ *is said a* normalization condition.

Proposition 1.3.1 is easily proved by induction on the length of the term $t(x)$ using the following lemmas.

Lemma 1.3.1 (Pseudo-inversion of P and S). *For any* x *we have* $PS(x)=x$. *For any* x *different from zero, we have* $SP(x)=x$. *For* $x=0$, *we have* $SP(x)=S(x)$.

Lemma 1.3.2 (Pseudo-commutation of P, S and K, L).
(1) If $K(x)\neq 0$ then $KS(x)=PK(x)$. If $K(x)=0$ then $KS(x)=SL(x)$.
(2) If $K(x)\neq 0$ then $LS(x)=SL(x)$. If $K(x)=0$ then $LS(x)=0$.
(3) If $L(x)\neq 0$ then $KP(x)=SK(x)$. If $L(x)=0$ then $KP(x)=0$.
(4) If $L(x)\neq 0$ then $LP(x)=PL(x)$. If $L(x)=0$ then $LP(x)=PK(x)$.

Proof. It suffices to write x as an expression of the form $C(a,b)$.
(1) (a) $KS(x)=KSC(a,b)=KC(a-1,b+1)=a-1=PK(x)$.
(b) $KS(x)=KSC(0,b)=KC(b+1,0)=b+1=SL(x)$.
(2) (a) $LS(x)=LSC(a,b)=LC(a-1,b+1)=b+1=SL(x)$.
(b) $LS(x)=LSC(0,b)=LC(b+1,0)=0$.
(3) (a) $KP(x)=KPC(a,b)=KC(a+1,b-1)=a+1=SK(x)$.
(b) $KP(x)=KPC(a,0)=KC(0,a-1)=0$.
(4) (a) $LP(x)=LPC(a,b)=LC(a+1,b-1)=b-1=PL(x)$.
(b) $LP(x)=LPC(a,0)=LC(0,a-1)=a-1=PK(x)$. □

1.4. Condition of elimination of quantifiers

From the previous discussion a necessary and sufficient condition for eliminating quantifiers is the following.

Proposition 1.4.1. *The theory* $Th(\mathbb{N},\mathscr{L})$ *eliminates quantifiers if, and only if, any formula of the form*

$$\exists x\left\{\left(\bigwedge_{i=1}^{n} M_i(x)=M_i'(x)+k_i\right)\wedge\left(\bigwedge_{i=n+1}^{m} M_i(x)\neq M_i'(x)+k_i\right),\right.$$
$$\wedge\left(\bigwedge_{i=m+1}^{p} M_i(x)=M_i'(y_{j_i})+k_i\right)\wedge\left(\bigwedge_{i=p+1}^{q} M_i(x)\neq M_i'(y_{j_i})+k_i\right),$$
$$\left.\wedge\left(\bigwedge_{i=q+1}^{r} M_i(y_{j_i})=M_i'(z_{j_i})+k_i\right)\wedge\left(\bigwedge_{i=r+1}^{s} M_i(y_{j_i})\neq M_i'(z_{j_i})+k_i\right)\right\},\qquad (F)$$

(*where the M_i's and the M_i''s are $\mathcal{N}$-terms, whose the set is supposed not to be empty, the y_{ji}'s and the z_{ji}'s being variables different from the variable x, the k_i's being positive, negative or null integers*) *is equivalent to a boolean combination of atomic $\mathcal{M}$-formulas just depending on the terms M_{ji} and M_{ji}' of the given formula.*

Proof. As usual when we want to eliminate quantifiers, it suffices to eliminate $\exists x$ within a formula of the form $\exists x\Phi(x, y_1, \ldots, y_n)$, where Φ is a conjunction of atomic or negatomic formulas. □

Actually the subformula $(\bigwedge_{i=q+1}^{r} M_i(y_{j_i}) = M_i'(z_{j_i}) + k_i) \wedge (\bigwedge_{i=r+1}^{s} M_i(y_{j_i}) \neq M_i'(z_{j_i}) + k_i)$ of this formula does not depend on x; it occurs just as a condition. We denote it by $CONDEXTOR$ (meaning *external original condition*).

In order to perform the reduction of a formula of the form (F), we shall use the representation of an $\mathcal{N}$-term by a word on the alphabet $\{K, L\}$ and the representation of an open $\mathcal{M}$-formula by a finite set of labelled trees in a specific way we precise later on.

2. Representations of $\mathcal{N}$-terms and open $\mathcal{M}$-formulas in one variable

2.1. Words associated to $\mathcal{N}$-terms

Let us note that an $\mathcal{N}$-term, say $t = K(K(L(L(K(x)))))$, is determined by a variable, namely x in the present case, and a word on the alphabet $\{K, L\}$, denoted[1] by $m(t)$, namely $KKLLK$ in the present case. This leads to the following.

Definition 2.1.2. (1) The *word* $m(t)$ on $\{K, L\}$ which is *associated* to the $\mathcal{N}$-term t is defined by structural induction as follows:
- if t is a variable then $m(t) = \lambda$, where λ is the empty word;
- if $t = K(t')$ then $m(t) = Km(t')$;
- if $t = L(t')$ then $m(t) = Lm(t')$.

(2) The *length* $lg(t)$ of the $\mathcal{N}$-term t is the length of the word $m(t)$.

2.2. Parametrized trees

With the previous definitions, $\mathcal{N}$-terms are represented by words of $\{K, L\}^*$, and now we are going to consider the specific trees we intend to associate to the investigated $\mathcal{M}$-formulas.

Definition 2.2.1. The $\mathcal{N}$-term $f_{n,i}(x)$, where x is a variable, n and i are natural integers, is defined by induction on n and i as follows:
- $f_{0,0}(x) = x$,

[1] The french equivalent to *word* is *mot*.

- $f_{n+1,2^n+j}(x) = f_{n,j}(L(x))$, for $0 \leqslant j < 2^n$,
- $f_{n+1,j}(x) = f_{n,j}(K(x))$, for $0 \leqslant j < 2^n$.

Example. For $n = 2$, the four possible terms $f_{2,i}(x)$ are

$$KK(x),\ KL(x),\ LK(x),\ LL(x),$$

since

$$\begin{aligned}
&f_{2,0}(x) = f_{1,0}(K(x)) = f_{0,0}(K(K(x) = KK(x),\\
&f_{2,1}(x) = f_{1+1,1}(x) = f_{1,1}(K(x)) = f_{0+1,2^0+0}(K(x)) = f_{0,0}(LK(x)) = LK(x)\\
&f_{2,2}(x) = f_{1+1,2^1+0}(x) = f_{1,0}(L(x)) = f_{0,0}(KL(x)) = KL(x),\\
&f_{2,3}(x) = f_{1+1,2^1+1}(x) = f_{1,1}(Lx) = f_{0,0}LL(x) = LLx.
\end{aligned}$$

Proposition 2.2.1. *For any nonnegative integer n, we have $x = y$ if and only if*

$$\forall i \in [0, 2^n - 1]\ (f_{n,i}(x) = f_{n,i}(y)).$$

Definition 2.2.2. For any variable x and any natural integer $n \in \mathbb{N}$, we put the notation

$$[Arb(x,n), (B_0, \ldots, B_{2^n-1})],$$

and we call *tree with conditional parametrized leaves*, or more briefly *parametrized tree*, any ordered pair such that

– The first component of the ordered pair is the binary tree with height $n + 1$ which is labelled by the $\mathcal{N}$-terms t with length $lg(t) \leqslant n$ in the usual way as shown in Fig. 1

– The second component of the ordered pair is an 2^n-tuple whose the ith coordinate is, for $0 \leqslant i \leqslant 2^n - 1$, a finite set B_i, we call *constraints box*, which consists of conditions (or constraints) of one of the following forms:

(1) either $t(f_{p,i}(x)) = m$;
(1′) or $t(f_{p,i}(x)) \neq m$;
(2) or $f_{p,i}(x) = v(f_{p,j}(x)) + m$;
(2′) or $t(f_{p,\,i}(x)) \neq v(f_{p,\,j}(x)) + m$;
(3) or $f_{p,i}(x) = v(y) + m$;
(3′) $t(f_{p,i}(x)) \neq v(y) + m$,

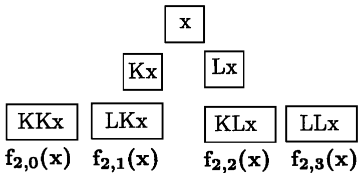

Fig. 1. $Arb(x, 2)$.

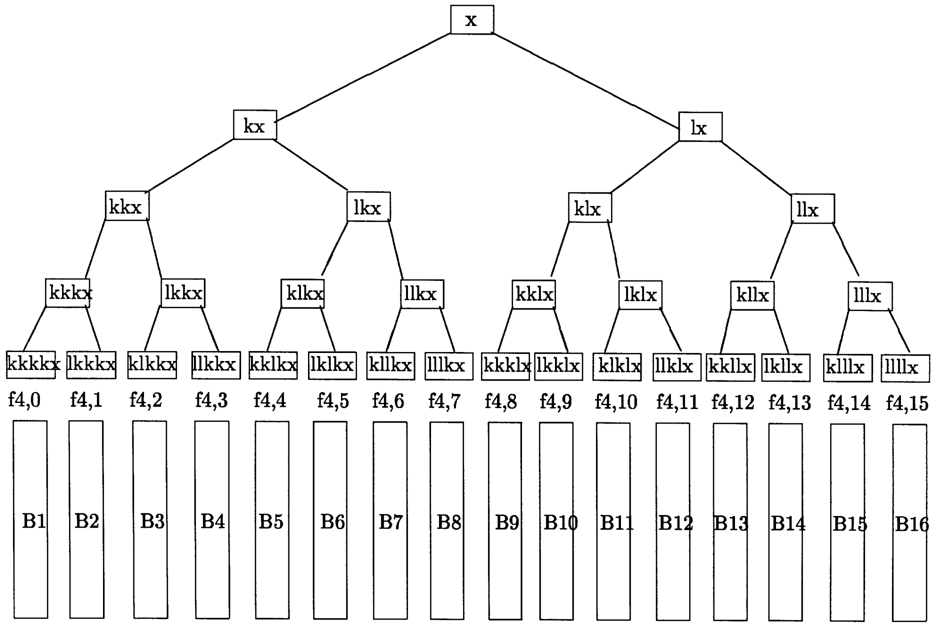

Fig. 2.

where $t(z)$ and $v(z)$ are $\mathcal{N}$-terms, where j is a natural integer which can be equal to i, where y is a variable different from x, and where m is an integer (necessarily positive if the word associated to $m(t)$ is empty).

We note point ($1'$) is a special case of point ($3'$). We consider ($1'$) purely to keep symmetry. Above we give (Fig. 2) an example of parametrized tree with height 4. In order to provide a convenient legibility, we place the box B_i exactly under the leaf $f_{p,i}(x)$.

Remark. The fact that there is no symmetry between the case of Eq. (2) and disequations ($2'$) on the one hand and, on the other between Eq. (3) and disequations ($3'$) within Definition 2.2.2 (a term t appears in a case and not in the other one) is voluntary: actually, we do not know exactly how to eliminate the prefix t in a disequation but, fortunately, we know how to perform it in equations. However, this fact of not knowing how to simplify disequations will not be as much constraining as it seems at the first glance.

Now, we classify variables occurring in a parametrized tree.

Definition 2.2.3. Let $[Arb(x,n),(p_0,\dots,p_{2^n-1})]$ be some parametrized tree. The variable x (root of the tree) is called the *main variable*. The variables $y_1,\dots,y_p$ different from x occurring or not occurring in the constraints boxes are called *box-variables.*

The main variable and the box-variables make up the whole set of *free variables* of the tree.

2.3. Semantics of parametrized trees

In the previous sections we introduced expanded languages and parametrized trees which are on a sense the syntactical framework of our proof of decidability of the theory $Th(\mathbb{N}, C, S)$. In order to justify this decision process on the set $\mathbb{N}$ of natural integers, we need to interpret all the introduced notions as trees, roots, variables, constraints, boxes. This is what we intend to do in the present section.

Definition 2.3.1. Let $T(x; y_1, \dots, y_q)$ be some parametrized tree with main variable x and box-variables $y_1, \dots, y_q$. The $(q+1)$-tuple $\langle a, b_1, \dots, b_q\rangle$ of natural integers *satisfies the parametrized tree* $T(x; y_1, \dots, y_p)$ if, and only if, for every constraints box B_i, all conditions, obtained from the constraints of this box by substitution of a to x and of the adequate b_j to each corresponding y_j are satisfied.

Definition 2.3.2. Let $T(x; y_1, \dots, y_q)$ be some parametrized tree with main variable x and box-variables $y_1, \dots, y_q$. Given the q-tuple $\langle b_1, \dots, b_q\rangle$ of natural integers the parametrized tree $T(x; b_1, \dots, b_p)$ is said to be *consistent* if, and only if, there is some natural integer a which satisfies this tree.

Definition 2.3.3. Let $\mathscr{F} = \{T_r(x; y_1, \dots, y_q)/r \in I\}$ and $\mathscr{G} = \{T_s(x; y_1, \dots, y_q)/s \in J\}$ be finite families of parametrized trees having the same main variable x. We say these *families* are *equivalent* if, and only if, for every q-tuple $\langle b_1, \dots, b_q\rangle$ of natural integers, the set of natural integers a such that $\langle a, b_1, \dots, b_q\rangle$ satisfies all trees of $\mathscr{F}$ is equal to the set of natural integers a such that $\langle a, b_1, \dots, b_q\rangle$ satisfies all trees of $\mathscr{G}$. A parametrized tree T is *equivalent* to a family $\mathscr{F}$ of parametrized if, and only if, families $\{T\}$ and $\mathscr{F}$ are equivalent.

Remark. Of course, one can arrange things in order to have the same set of box-variables: it suffices to take the set of all occurring variables.

3. Solving some equations

In order to eliminate quantifiers, we are led to solve certain equations in which occur K and L. More precisely, we prove

Proposition 3.1.1. *The equation* $Mz = z - k$, *where* $M \in \{K, L\}^*$ *is a word of a non-zero length* $lg(M)$ *and* $k \in \mathbb{N}^*$, *has a finite number of solutions, all of them being bounded by a constant just depending on* k *and* $lg(M)$.

To prove Proposition 3.1.1, we shall use several lemmas.

Lemma 3.1.1. *For all $z \geqslant 7$, the inequality $K(z) < z/2$ holds.*

Proof. Let z be an arbitrary natural integer. There is a unique natural integer d such that

$$\frac{d(d+1)}{2} \leqslant z < \frac{(d+1)(d+2)}{2}.$$

This integer d is nothing but the number of the diagonal to which belongs point $\langle K(z), L(z)\rangle$, whose Cantor number is z, so that $K(z)+L(z)=d$. Since we have

$$\frac{(d+1)(d+2)}{2} - \frac{d(d+1)}{2} = d+1,$$

there is a unique integer r such that

$$z = \frac{d(d+1)}{2} + r$$

verifying $0 \leqslant r < d+1$.

Then we have $L(z)=r$, $K(z)=d-r$, hence $K(z) \leqslant d$.

But $d(d+1)/2 + r = z$ leads to $d^2 + d + 2r = 2z$, hence $d^2 < 2z$, so that $d < \sqrt{2z}$; so we have $K(z) > \sqrt{2z}$. If $\sqrt{2z} \leqslant z/2$, then $K(z) < z/2$. By squaring $\sqrt{2z} < z/2$ we see this is the case for $z \geqslant 9$.

One easily checks that, for $z \leqslant 8$, the required strict inequality holds, except for $z=0$ and $z=6$, achieving the proof. □

Lemma 3.1.2. *For all $z \geqslant 3$, the inequality $L(z) < z/2$ holds.*

Proof. Putting once more z under the form $z = d(d+1)/2 + r$, we have $L(z)=r$. Hence

$$L(z) = r \leqslant d < \frac{z}{2} \quad \text{for } z \geqslant 9,$$

according to what we just did in the proof of previous lemma. One easily checks that, for $z \leqslant 8$, the requested strict inequality holds, except for $z=0$ and 2, achieving the proof. □

Proof of Proposition 3.1.1. Let M be a word with a non-zero length. Let a be an integer.

- If there is no suffix M' of M such that $M'(a) \leqslant 6$, then we have $M(a) < a/2^n$ according to previous Lemmas 3.1.1 and 3.1.2.
- If there is a suffix M' of M such that $M'(a) \leqslant 6$, then we have $M(a) \leqslant 6$ so that $M(a) < a/2^n$, hence we get $M(a) < a/2^n$ for $a > 6.2^n$.
- Of course, there are finite solutions in the equation $M(z) = z - k$ for $z \leqslant 6.2^n$.
- For $a > 6.2^n$, we have $M(a) < a/2^n$. Hence $M(z) = z - k$ implies $z < k.2^n/(2^n - 1)$.
- Then it suffices to search for solutions within a finite set. □

Proposition 3.1.2. *The set of solutions of the equation $Mz = z$, where $M \in \{K, L\}^*$ is a word of non-null length, is included in $\{0, 1\}$.*

Proof. More precisely, 0 is always a solution and 1 is a solution if and only if $M = K^n$. For $z \geqslant 2$, we have $M(z) < z$ and hence $M(z) \neq z$. $\square$

Proposition 3.1.3. *The equation $Mz = z + k$, where $M \in \{K, L\}^*$ and $k \in N^*$, has no solution.*

Proof. Indeed we have $M(z) \leqslant z$, consequently $M(z) \neq z + k$. $\square$

4. Reduction of a formula to a family of trees

We have to eliminate quantifiers from an existential formula. We transform this existential formula in x into a finite set (we say *family*, which are below always finite) of trees, whose representation as a disjunction of open formulas is easy, and a set of external conditions where x does not occur.

Proposition 4.1.1. *Let Φ be a formula of the form* (F) *of the Section* 1.4. *There exists an $\mathscr{M}$-formula* CONDEXTOR *and a* (*finite*) *family $\mathscr{F}$ of ordered pairs*, *each of them consisting of a parametrized tree T and of an $\mathscr{M}$-formula* CONDEXT(T), *such that the formula Φ is equivalent to a formula denoted by* CONDEXT, *with no occurrence of x and which is the conjunction of the formula* CONDEXTOR *with the disjunction of all formulas* CONDEXT(T) *that are the first components of an ordered pair belonging to $\mathscr{F}$ whose associated tree is consistent.*

Proof. The existence of such a family is proved in a constructive way. The (finite) family of the ordered pairs associated to Φ will be obtained at the last step of the long process which follows in sections up to Section 7 below. At each step of the process, we shall deal with cases and subcases and in every case or subcase we call T the tree to be transformed and T' the transformation obtained tree.

Initialization step: We refer to formula (F) of Section 1.4 with its literals and its bounds n, m, p, q, r, s. The formula *CONDEXTOR* has been defined in Section 1.4. Let N be the maximum of lengths of $\mathscr{N}$-terms in x occurring in the formula Φ. Let k_0 be the maximum of absolute value of integer constants occurring in the first m literals of formula (1). We start by considering the family of ordered pairs $(T,\ CONDEXT(T))$, where

- T is one of the $(k_0 + 1)^{2^N}$ trees of height N whose every box B_i contains one of the $k_0 + 1$ conditions:
- either $f_{N,i}(x) = \alpha$, where α is an integer constant lesser than k_0,
- or $f_{N,i}(x) \geqslant k_0$ (of course it is an abbreviation for the k_0 conditions $f_{N,i}(x) \neq 0$, $f_{N,i}(x) \neq 1, \ldots, f_{N,i}(x) \neq k_0 - 1$);

– $CONDEXT(T)$ contains, for each $i \in [m+1, q]$, for each $\mathcal{N}$-terms W of length less than $N - length(M_i)$, a factor:
– either $WM'_i(y_{j_i}) = \beta$, where β is an integer constant lesser than $|k_i|$,
– or $WM'_i(y_{j_i}) \geqslant |k_i|$.
This family represents all possibilities of ordered pairs $(T, CONDEXT(P))$ up to an obvious isomorphism.

Equivalent step: Now we treat every tree of this family, i.e. we add conditions in boxes to obtain a set of conditions equivalent to the first q literals of formula (1). Let T be such a tree. We shall inspect every literal and odd conditions. Treatment depends on the form of literal.

Case 1: At first, we deal with the n literals of the form $M(x) = M'(x) + k$ (atomic formulas). We must distinguish two kinds of literals among the literals of this form according to the fact whether M is a suffix of M'.

Case 1.1: Let us begin by looking at the literals of the second kind, namely those of the form $M(x) = M'(x) + k$ where neither M nor M' is a suffix of each other, what implies that *the subtrees having respective roots* $M(x)$ and $M'(x)$ *are disjoint*. Moreover, one can assume by symmetry that $lg(M(x)) \geqslant lg(M'(x))$.

Since, the length of the tree is greater than N, there exists two nodes of T (which are not necessarily leaves) labelled $M(x)$ and $M'(x)$. For every leaf $f_{N,i}(x)$ of the subtree having $M'(x)$ for root, there exist a word, say U, on the alphabet $\{K, L\}$ such that we have $f_{p,i}(x) = UM'(x)$. The equality $M(x) = M'(x) + k$ is equivalent (due to Proposition 2.2.1) to the set of conditions $UM'(x) = U(M(x) - k)$ for leaves of the subtree of root $M'(x)$. The initial conditions of tree T lead to a normalization

$$UM'(x) = V_i V'_i M(x) + m_i,$$

where V_i and V'_i are words on $\{K, L\}$, where $lg(V'_i M) = lg(UM')$ and m_i is an integer.
The node $V'_i M(x)$ is a leaf $f_{N,j}(x)$ of the subtree having $M(x)$ for root.
Therefore we have to add constraints

$$f_{N,i}(x) = V_i(f_{N,j}(x)) + m_i$$

in box B_i. Because the subtrees are disjoint, we see that leaves $f_{p,i}(x)$ and $f_{p,j}(x)$ are different. We note that if V is the empty word and if m is strictly negative then we can forget the corresponding tree since one is sure it cannot be satisfied due to the fact that $f_{N,i}(x)$ cannot take any negative value.

Case 1.2. Let us now consider the case when M' is a suffix of M, i.e. the case when $M = M''M'$ for some M''. Put $\tilde{x} = M'(x)$. Therefore the literal $M(x) = M'(x) + k$ is equivalent to $M''(\tilde{x}) = \tilde{x} + k$. According to Section 3, the two following subcases arise.

Subcase 1.2.1: The word M'' is empty ($M'' = \lambda$). If $k = 0$ then there is no constraint and we do not have to add any condition in the box. If $k \neq 0$ there is a contradiction and we may forget the considered tree.

Subcase 1.2.2: The word M'' is not empty ($M'' \neq \lambda$). In this case, according to Section 3, the solutions in $\tilde{x}$ are taken in the finite set $\{\alpha_1, \ldots, \alpha_Q\}$ of solutions of the equation $M''(\tilde{x}) = \tilde{x}+k$. First, every tree T is cloned into Q trees $T_1, \dots, T_Q$. Afterwards, for the tree T_j, $1 \leqslant j \leqslant Q$, we add to the constraints boxes B_i, where $f_{N,i}(x)$ is a leaf of the subtree having $\tilde{x}$ as its root, an extra constraint $f_{N,i}(x) = m_i$. The value of the constant m_i depends at the same time on the value α_j and of the path going from $\tilde{x}$ to the considered leaf. More precisely if $f_{N,i}(x) = M'''(\tilde{x})$ then $m_i = M'''(\alpha_j)$.

Case 2: We deal with literals of the form $M(x) \neq M'(x) + k$ (*negatomic formulas*) in a similar way, by denying what we get in case of equalities.

Case 3: Let us deal with literals of the form $M(x) = M'(y) + k$. Since the tree T has height N, which is the maximum of lengths of the terms occurring in the formula, there exists in the tree some nodes labelled by the name $M(x)$. For every leaf $f_{N,i}(x)$ of the subtree having $M(x)$ for its root, there exists a word U on $\{K, L\}$ such that $f_{N,i}(x) = UM(x)$.

The equality $M(x) = M'(y) + k$ is equivalent to the set of equalities

$$UM(x) = U(M'(y) + k)$$

for all the leaves of the subtree of root $M(x)$. The formula $CONDEXT(T)$ contains conditions for normalization allowing us to replace the above formula by

$$UM(x) = VM'(y) + m'$$

for an $\mathcal{N}$-term V of same length than U and an integer constant m'.

We add the constraint $f_{p,i}(x) = VM'(y) + m'$ in the box B_i of the tree T.

Case 4: In a similar way, one can deal with conditions of the form $M(x) \neq M'(y) + k$. $\square$

Remark. At the end of this first step, the boxes of constraints of the considered parametrized trees contain constraints of the form (F) with $lg(t) \leqslant N$, (1′), (2) or (2′) with $j \neq i$, (3) or (3′).

Conclusion: To finish elimination of quantifiers, we have to determine consistency of parametrized trees. To determine the consistency of a parametrized tree obtained at the end of this first step of reduction seems a little bit easier than eliminating quantifier from formula (F). Indeed, for every constraints box, it suffices to find a natural integer $f_{N,i}(x)$ verifying some constraints of the forms we recall above.

It is easy to see whether a set of conditions of forms (1), (1′), (3) and (3′) are compatible, even if we must add for doing this some factors in the external disjunction. Only the conditions
(2) $f_{N,i}(x) = v(f_{N,j}(x)) + m$,
(2′) $f_{N,i}(x) \neq v(f_{N,j}(x)) + m$;
provide some problems. This is due to two reasons: on the one hand, there is no intrinsic order on the leaves such that if we have (2) then we have $f_j < f_i$ (what can lead to viceous circle in the algorithm); on the other, even in case there does exist such an intrinsic order, it would remain some latitude to assign a value to certain leaves so

that the appearance of some uncompatibility at the level of some leaves would result of a wrong choice and not at all of the inconsistency of the tree.

5. Extension of a parametrized tree

To solve the problem pointed above, we shall have to expand a given tree into a finite set of trees with increasing heights. Here, we must emphasize on the following point: we cannot a priori bound the size of our trees. We have to exend several times successively the trees in order to obtain a family of simpler trees *w*hose consistency is easy. Now, we are going to define what we call *extension* of the parametrized tree $[Arb(x,N),(B_0,\dots,B_{2^N-1})]$ in the family of trees $[Arb(x,M),(C_0,\dots,C_{2^M-1})]$ for $M>N$. It suffices to define this notion in the case $M=N+1$ and for doing this, due to the fact the integer N and the variable x completely determine the underlying binary tree, it suffices to determine the new constraint boxes of the trees which expand the given tree.

Definition 5.1.1. An *extension of a parametrized tree T* of height N is a finite family of parametrized tree of height $N+1$ which is equivalent to T (here considered as a family consisting of just one element).

Remark. Every tree belonging to the extension has a number of constraints boxes which is double the number of constraints boxes of T.

Proposition 5.1.1. *Any extension of a parametrized tree T of height N is the finite set of parametrized trees of height* $N+1$ *defined by conditions* (1)–(7) *below.* (*We denote by adjunction of a symbol prime* ($'$) *the constraints boxes concerning the trees of the extension of T, so that a box* B_i *of the original tree gives rise to two new boxes* B'_{2i-1} *and* B'_{2i} *for every tree of the extension of* T).

Transformations due to equations:

(1) *If* B_i *contains* $t(f_{N,i}(x))=\alpha$ *then we distinguish the following three cases:*
- *if t is the empty word then* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=\beta$ *and* B'_{2i} *contains* $f_{N+1,2i}(x)=\gamma$, *where* β *and* γ *are integer constants such that* $K(\alpha)=\beta$ *and* $L(\alpha)=\gamma$;
- *if* $t=uK$, *with* $u\in\{K,L\}^*$, *then* B'_{2i-1} *contains* $u(f_{N+1,2i-1}(x))=\alpha$;
- *if* $t=uL$, *with* $u\in\{K,L\}^*$, *then* B'_{2i} *contains* $u(f_{N+1,2i}(x))=\alpha$.

(2) *It suffices to explain the case when* B_i *contains* $f_{N,i}(x)=t(f_{N,j}(x))+m$ *on an example. Let us suppose* $f_{N,i}(x)=t(f_{N,j}(x))+2$ *and let us just consider the situation of the box* B'_{2i-1}.
- *If* $t=\lambda$ *is the empty word, we obtain the following three families of trees*:
  - *a family for which* B'_{2j-1} *contains* $f_{N+1,2j-1}(x)=0$ *and* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=f_{N+1,2j}(x)$;

– *a family for which* B'_{2j-1} *contains* $f_{N+1,2j-1}(x)=1$ *and* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=f_{N+1,2j}(x)+2$;
– *a family for which* B'_{2j-1} *contains* $f_{N+1,2j-1}(x)\geqslant 2$, *and* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=f_{N+1,2j}(x)-2$.

- *If* $t=uK$, *with* $u\in\{K,L\}^*$, *then this gives rise to three families of trees*:
  – *a family for which* B'_{2j-1} *contains* $Ku(f_{N+1,2j-1}(x))=0$ *and* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=Lu(f_{N,2j-1}(x))$;
  – *a family for which* B'_{2j-1} *contains* $Ku(f_{N+1,2j-1}(x))=1$ *and* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=Lu(f_{N+1,2j-1}(x))+2$;
  – *a family for which* B'_{2j-1} *contains* $Ku(f_{N+1,2j-1}(x))\geqslant 2$ *and* B'_{2i-1} *contains* $f_{N+1,2i-1}(x)=Lu(f_{N+1,2j-1}(x))-2$.

(3) *If* B_i *contains* $f_{N,i}(x)=v(y)+k$ *then* B'_{2j-1} *contains* $f_{N+1,2i-1}(x)=K(v(y)+k)$ *and* B'_{2i} *contains* $f_{N+1,2i}(x)=L(v(y)+k)$.

Transformations due to disequations:
We deal with disequations in a similar way by considering negations of the conditions we just obtained. For instance if B_i *contains* $f_{N,i}(x)\neq\alpha$ *then two families of extended trees appear, in one of them* B'_{2i-1} *contains* $f_{N+1,2i}(x)\neq\beta$ *and in the other* B'_{2i} *contains* $f_{N+1,2i}(x)\neq\gamma$ *with* $K(\alpha)=\beta$ *and* $L(\alpha)=\gamma$.

Proof. To justify points (1) and (2) it suffices to note that the sons (relatively to K and L respectively) of the leaf $f_{p,i}(x)$ are the respective leaves $f_{N+1,2i-1}(x)$ and $f_{N+1,2i}(x)$.
(1) (a) The case when $t=\lambda$ is the empty word is trivial.
(b) If $t=uK$, then $t(f_{N,i}(x))=m$ is equivalent to $uK(f_{N,i}(x))=m$, but we have $K(f_{N,i}(x))=f_{N+1,2i-1}(x)$, proving the result.
(2) (a) For the case when $t=\lambda$ is the empty word, we make use of the given rules defining our terms normalization,
- if $K(f_{N,j}(x))=0$ then $K(f_{N,j}(x)+1)=L(f_{N,j}(x))+1$.
- if $K(f_{N,j}(x))\neq 0$ then $K(f_{N,j}(x)+1)=K(f_{N,j}(x))-1$ and $L(f_{N,j}(x)+1)=L(f_{N,j}(x))+1$.

Hence,
- if $K(f_{N,j}(x))=0$ then $K(f_{N,j}(x)+1)=L(f_{N,j}(x))+1\neq 0$, hence $K(f_{N,i}(x))=L(f_{N,j}(x))+1-1=L(f_{N,j}(x))$.
- if $K(f_{N,j}(x))=1$ then $K(f_{N,j}(x)+1)=K(f_{N,j}(x))-1=0$, hence $K(f_{N,i}(x))=L(f_{N,j}(x)+1)+1=L(f_{N,j}(x))+2$.
- if $K(f_{N,j}(x))\geqslant 2$ then $K(f_{N,j}(x)+1)=K(f_{N,j}(x))-1\neq 0$, hence $K(f_{N,i}(x))=K(f_{N,j}(x)+1)-1=K(f_{N,j}(x))-2$.

For the extended trees,
- in the first case, we have $f_{N+1,2j-1}(x)=0$ and $f_{N+1,2i-1}(x)=f_{N+1,2j}(x)$,
- in the second case, we have $f_{N+1,2j-1}(x)=1$ and $f_{N+1,2i-1}(x)=f_{N+1,2j}(x)+2$,
- in the third case, we have $f_{N+1,2j-1}(x)\geqslant 2$ and $f_{N+1,2i-1}(x)=f_{N+1,2j-1}(x)-2$.

(b) The case when $t=uK$ is analogous and can be developed as follows for the considered example:

- if $KM(f_{N,j}(x)) = 0$ then $K(M(f_{N,j}(x)) + 1) = LM(f_{N,j}(x)) + 1$,
- if $KM(f_{N,j}(x)) \neq 0$ then $K(M(f_{N,j}(x)) + 1) = KM(f_{N,j}(x)) - 1$ and $L(M(f_{N,j}(x)) + 1) = LM(f_{N,j}(x)) + 1$.

Hence,

- if $KM(f_{N,j}(x)) = 0$ then $K(M(f_{N,j}(x)) + 1) = LM(f_{N,j}(x)) + 1 \neq 0$, hence $K(f_{N,j}(x)) = LM(f_{N,j}(x)) + 1 - 1 = LM(f_{N,j}(x))$.
- if $KM(f_{N,j}(x)) = 1$ then $K(M(f_{N,j}(x)) + 1) = KM(f_{N,j}(x)) - 1 = 0$, hence $K(f_{N,j}(x)) = LM(f_{N,j}(x)) + 2$.
- if $KM(f_{N,j}(x)) \geqslant 2$ then $K(M(f_{N,j}(x)) + 1) = KM(f_{N,j}(x)) - 1 \neq 0$, hence $K(f_{N,j}(x)) = KM(f_{N,j}(x)) - 2$.

This can be translated, for the extended trees in the case of our example where $M = M'K$, using the fact that $Kf_{N,j}(x) = f_{N+1,2j-1}(x)$, by

- in the first case, $KM'(f_{N+1,2j-1}(x)) = 0$ and $f_{N+1,2i-1}(x) = LM'(f_{N+1,2j-1}(x))$,
- in the second case, $KM'(f_{N+1,2j-1}(x)) = 1$ and $f_{N+1,2i-1}(x) = LM'(f_{N+1,2j-1}(x)) + 2$,
- in the third case, $KM'(f_{N+1,2j-1}(x)) \geqslant 2$ and $f_{N+1,2i-1}(x) = KM'(f_{N+1,2j-1}(x)) - 2$.

All the other cases are treated in an analogous way. $\square$

6. Simplification of parametrized trees

Up to now, what we did consists in associating a family of ordered pairs (namely each of them consists of a formula and a tree) to the original formula (F) we intend to eliminate the existential quantifier. Therefore, the question on the original formula is transfered to a problem of consistency of trees. The first step of the determination of the consistency of a parametrized tree is devoted to associate to a fixed tree T a new family of more simple (in a sense) trees which is equivalent to T.

6.1. Simple tree

The notion of a *simple tree* is related to the content of the constraints boxes of the considered tree. Let us begin by noting that the (of course finitely many) constraints of a tree can be classified in several types we introduce in the following definition.

Definition 6.1.1. (1) A *fundamental constraint* is a constraint of one of the following forms:

(1) $t(f_{N,i}(x)) = m$ with t having a non-null length;
(1′) $t(f_{N,i}(x)) \neq m$;
(2′) $t(f_{N,i}(x)) \neq t'(f_{N,k}(x)) + m$ (with the possibility k and i are the same index);
(3′) $t(f_{N,i}(x)) \neq v(y) + m$.

(2) We call *determining constraint* any constraint of the form $f_{N,i} = v(y) + m$ or $f_{N,i} = m$.

(3) We call *dependence constraints* any constraint of the form $f_{N,i} = t(f_{N,j}) + m$, where the leaf $f_{N,j}$ is different from the leaf $f_{N,i}$.

The notion of a *simple tree* lies on the nature and the number of the permitted constraints one can find within boxes. More precisely, in order to be simple a tree must satisfy conditions as in the following definition.

Definition 6.1.2. We call a *simple tree* any parametrized tree whose every constraints box contains:
(a) either nothing;
(b) or only one determining constraint,
(c) or only one constraint of dependence $f_i = t(f_j) + m$,
(d) or only a set of more than one determining constraints.

The main result of section Section 6 is completely contained in the next proposition.

Proposition 6.1.1. *Every parametrized tree is equivalent to a family of simple trees.*

Proof (*Sketch*)**.** Let T be a parametrized tree. By the very definition of such a tree, a constraints box B_i of this tree contains a finite set (possibly empty) of constraints which are necessarily of one of the following forms:
(1) $t(f_{N,i}(x)) = m$;
(1′) $t(f_{N,i}(x)) \neq m$;
(2) $f_{N,i}(x) = v(f_{N,j}(x)) + m$;
(2′) $t(f_{N,i}(x)) \neq v(f_{N,k}(x)) + m$;
(3) $f_{N,i}(x) = v(y) + m$;
(3′) $t(f_{N,i}(x)) \neq v(y) + m$.
Now, the reduction to a family of simple trees is performed via an exhaustive classification of the contents of the boxes as follows.

- The content of a box is said to be of type α if there exists in the considered constraints box, a *determining constraint*, say $f_i = v(y) + m$.
- The content of a box is said to be of type β if there does not exist in the considered constraints box any determining constraint but there does exist a constraint of form (2), say $f_{N,i}(x) = v(f_{N,j}(x)) + m$, where the length of v is different from zero.
- The content of a box is said to be of type γ if there does not exist in the considered constraints box any determining constraint but there does exist one or several constraint of form (2), where the lengths of the terms v are different from zero.
- The content of a box is said to be of type δ if there exist in the considered constraints box neither a determining constraint nor a constraint of the form (2).

If a constraints box has a content of

- type α then we are going to show in Section 6.3 below some intuitive result from which we shall see that the corresponding leaf f_i is completely determined (modulo some y) so that the other constraints in the box of the same leaf can be advantageously carried in another constraints box without involving anymore the leaf f_i; afterwards we shall be in case (b) of the definition of simple trees;

- type β then we are going to show, what is also an intuitive strategy, that it suffices to keep only one such constraint; afterwards we shall be placed in case (c) of the definition of simple trees;
- type γ, then we shall show that we are once again led to case (c) of the definition of simple trees;
- type δ, then we shall show that we are led to case (a) or (d) of the definition of simple trees. □

6.2. Equivalent family of saturated trees

Let us note a constraints box B_i can apparently seem not to be of the type α although it contains some constraints $f_i = v(f_j) + m$ where the box B_j is of type α. Of course, the box B_i is "morally" speaking of this type α. For this reason we wish to consider a special kind of trees which are saturated in the sense we explained below.

Definition 6.2.1. A *parametrized tree* is said to have *saturated boxes*, and is called a saturated tree, when, for every pair of boxes B_j and B_i, if we have the constraint $f_i = v(f_j) + m$ then:

- if $t(f_j) = q$ is in the box B_j, and if t is a suffix of v, that is to say if v can be written as $v = v't$, then we have $f_i = v'(q) + m$ in the box B_i;
- if $f_j = q$ is in the box B_j and if t is not a suffix of v, then we do not change anythings;
- if $f_j = t(f_k) + r$ is in the box B_j then we have one of the ordered pair of normalization $u(f_k) = q$ (or $u(f_k) \geqslant q$) and $f_i = v'(t(f_k)) + r'$ in the boxes B_k and B_i;
- if $f_j = t(y) + r$ is in the B_j then we have one of the external condition $u(y) = q$ (or $u(y) \geqslant q$) and $f_i = v'(t(y)) + r'$ in the box B_i.

We begin by proving our notion of saturation of trees is conservative for the truth.

Lemma 6.2.1. *Every parametrized tree is equivalent to a family of trees whose constraints boxes are saturated* (*saturated trees*).

Proof. If the constraint $f_i = v(f_j) + m$ is contained in the box B_i then one adds, if they are not already there, the adequate constrainsts.

We start again until we cannot add any more constraints, so that all boxes are saturated. Let us notice that this process terminates since when there are Q leaves and the maximum number of constraints is R in a given box then; after performing as above, the box containing the maximum number of constraints will contain at most QR constraints.

Since, during the process, the conjunction of all the constrainsts of all boxes remains invariant, we get an equivalent tree. □

One can note that, after each step of simplification, the obtained trees are such that it is useless to begin again this process of saturating boxes (although one could perform it easily without any problem).

6.3. Equivalent family of α-reduced trees

We are going to show that every leaf f_i whose constraints box contains at least a determining constraint $f_i = v(y) + m$ is completely determined (modulo some y), so that the other constraints due to this leaf may advantageously sent into either the external condition or one another constraints box (without involving the leaf f_i). We will prove that every parametrized tree is equivalent to a family of trees of those kind more precisely.

Definition 6.3.1. Any *parametrized tree* is called *α-reduced* iff all its constraints boxes containing at least a determining constraint consists of a singleton whose the unique element is the determining constraint.

This definition allows us to begin the proof we just announced.

Proposition 6.3.1. *Any parametrized tree is equivalent to a family of α-reduced trees.*

Proof. Let B_i be a constraint box containing both together a determining constraint and other constraints. Notice that several other determining constraints within B_i can appear.

Let us call *main determining constraint* of this box B_i the unique determining constraint of the form $f_i = v_0(y_0) + m_0$, where v_0 is the smallest term with respect to the lexicographical order on $\{K, L\}^*$ (among the determining constraints occurring in B_i), and y_0 is the smallest variable (for the usual order on variables) corresponding to that term if there are several such variables; if, after the previous process, there are still many determining constraints, then we shall select the unique determining constraint having a minimum m_0.

Now, consider in order of appearance all the constraints of the box B_i, taking them one after the other according to their form.

Case 1: Let us begin by constraints of form (1), namely $t(f_i) = q$. Such a constraint is equivalent to $t(v_0(y_0) + m_0) = q$. Therefore, the ordered pair $\langle T, CONDEXT(T)\rangle$ is equivalent to the ordered pair $\langle T', CONDEXT(T')\rangle$ where T' is obtained from T by removing the constraint $t(f_i) = q$ and $CONDEXT(T')$ is the conjunction of $CONDEXT(T)$ and $t(v_0(y_0) + m_0) = q$.

A constraint of form (1′), namely $t(f_i) \neq q$, can be treated in a similar way.

Case 2: Let us deal with constraints of form (2), namely equations of the form $f_i = w(f_j) + r$. Due to our hypothesis on f_i, such a constraint is equivalent to $v_0(y_0) + m_0 = w(f_j) + r$, still equivalent to $w(f_j) = v_0(y_0) + m_0 - r$.

The idea of the proof consists in putting a new constraint in the box B_j. So doing nevertheless, we do not obtain a parametrized tree since this constraint is alike a constraint of form (3) but is not necessarily of form (3) in reason of the occurrence of the term w.

If the constraint $w(f_j) = v_0(y_0) + m_0 - r$ is of form (3), in other words if w is the empty word then one removes the constraint $f_i = w(f_j) + r$ from B_i and we put the constraint $f_j = v_0(y_0) + m_0 - r$ into B_j. If the leaf f_j is determined then it suffices to put $m' = v_0(y_0) + m - r$ within $CONDEXT(T)$.

If the constraint $w(f_j) = v_0(y_0) + m_0 - r$ is not of the form (3) then the strategy of simplification lies on the fact of extending the tree into a family of trees (since there are several distinct cases in the process of normalization) having a height which is equal to the length c of w. This leads to get a constraint of the form $f_{p+c,j}(x) = v'(y_0) + m'$, what brings us back to previous case.

Case 2$'$: Let us deal with constraints of form (2), namely disequations of the form $t(f_i) \neq w(f_j) + r$. Due to our hypothesis on f_i, such a constraint is equivalent to $w(f_j) \neq t(v_0(y_0) + m_0) - r$. After normalization, this last constraint A is of form (3$'$). We put A in the adequate box after having removed from the box B_i the constraint which gave rise to A. If f_j is a determined leaf then we put $m' = t(v_0(y_0) + m_0) - r$ into $CONDEXT(T)$.

Cases 3 *and* 3$'$: If, in the box B_i we have external constraints (i.e. which bring on other variables than x, that is to say constraints of form (3) as $f_i = u(z) + r$ or (3$'$) as $w(f_i) \neq u(z) + r$) then we remove these constraints from the box B_i and we add within the formula $CONDEXT(T)$ as many subformulas of the forms $v_0(y_0) + m_0 = u(z) + r$ and $w(v_0(y_0) + m_0) \neq u(z) + r$ as necessary.

After this process, all the constraints boxes which contain a determining constraint does not contain any other determining constraint than this one. □

6.4. Equivalent family of α-β-1-reduced trees

Introduction: Now we deal with the simplification of the constraints boxes containing some constraint of the form $f_i = f_j + m$. We are going to show that if $i > j$ then the box B_i can be reduced to this unique constraint.

Definition 6.4.1. A *parametrized tree* is α-β-1-reduced iff, on the one hand, every constraints box containing a determining constraint is reduced to a singleton just containing such a constraint and, on the other, every constraints box containing a constraint of the form $f_i = f_j + m$ is reduced to a singleton just containing such a constraint.

This last Definition 6.4.1 allows us to begin the proof of the result we announced just above.

Proposition 6.4.1. *Any parametrized tree is equivalent to a family of* α-β-1-*reduced trees.*

Proof. According to Section 6.3, one can be satisfied with considering only α-reduced trees. Now let B_i be a constraint box containing both together a constraint of the form $f_i = f_j + m$ and other constraints. Assume, for instance, $i > j$.

Case 1: If the leaf f_j is determined then, due to saturation, it is the same for the leaf f_i and we come back to Section 6.3.

Case 2: There are in B_i and B_j several constraints of the form $f_i = f_j + m$ or $f_j = f_i - m$. If the values of m are all equal, then we remove these constraints from the box B_j and we keep only one constraint within B_i; so doing, we come back to the third case. Otherwise, the tree is not consistent and we can give it up.

Case 3: In this case neither f_i, nor f_j is determined and there is only one constraint which is equivalent to $f_i = f_j + m$ and lies in the box B_i. Let us put $C = B_i \cup \{f_i = f_j + m\}$. So we get a family (of trees T') equivalent to T by replacing B_i by $\{f_i = f_j + m\}$ and the constraints C as follows.

Subcase 1: If C contains the constraint $t(f_i) = q$, where t has a non-zero length then we have $t(f_j + m) = q$ and hence, by normalization, finitely many constraints of forms (1) and (1′) to put into B_j.

One treats constraints of form (1′) similarly.

Subcase 2: If C contains the constraints $f_i = w(f_k) + r$, with w of non-zero length then we put $f_j = w(f_k) + r - m$ into B_j.

Subcase 2′: If C contains the constraint $t(f_i) \neq w(f_k) + r$ then we see that $t(f_j + m) \neq w(f_k) + r$. We are led to add finitely many constraints of forms (1) and (1′) and a constraint of form (2′) in B_j.

Subcase 3′: If C contains the constraint $t(f_i) \neq w(y) + r$ then we see that $t(f_j + m) \neq v(y) + r$. We are led to add finitely many constraints of forms (1) and (1′) and a constraint of form (3′) in B_j.

During this process, we do not add any constraint in the box of a determined box and, consequently, the trees we obtain are α-reduced. We add in leaves f_j more constraints of forms (1), (1′), (2) and (3′). The only precaution we must take for avoiding to loop is to beginning by dealing with the leaves f_i where i is maximum. □

After α-β-1-reduction, any constraints box contains:
– either nothing,
– or a determined constraint,
– or a determined dependence constraint, say $f_i = f_j + m$, with $j < i$,
– or constraints of forms (1) with t of non-zero length, (1′), (2′), (3′) and (2) as $f_i = v(f_j) + m$ with v of non-zero length and where the leaf f_j is not determined and contains no constraint of the form $f_j = f_k + r$.

6.5. Equivalent family of α-β-2-reduced trees

Introduction: Now we deal with the simplification of the constraints boxes containing some constraint of form (2) as $f_i = v(f_j) + m$ with v of non-zero length where the leaf f_j is not determined and the associated box contains no constraint of the form $f_j = f_k + r$.

Definition 6.5.1. A *parametrized tree* is called α-β-2-*reduced* iff it is α-β-1-reduced and every constraints box of this tree containing a constraint of form (2), with a non-null term v, is reduced to a singleton just containing such a constraint.

The aim of the present section is to prove the following result.

Proposition 6.5.1. *Any parametrized tree is equivalent to a family of α-β-2-reduced trees.*

The fact that we can have several constraints boxes containing constraints of form (2) imposes to find an adequate strategy in order to avoid any looping situation. In this purpose, we introduce two complexity measures for a tree.

Definition 6.5.2. *The first complexity measure* of a parametrized tree T, we denote by $mes_1(T)$, is the length p which is minimum (in the set of all boxes of the tree) such that there exists a constraint of form (2), say $f_i = v(f_j) + m$, and another constraint of form (2), say $f_i = u(f_k) + s$, with v is a term of length p and u a term of length $\geqslant p$. This measure will be equal to 0 if there does not exist any box containing such a pair.

Remark. We have $mes_1(T) = 0$ if, and only if, every constraints box B_i of T contains within its subset of constraints of form (2), finitely many (eventually zero) constraints of the form $f_i = f_j + r$. We have seen in Section 6.4 how to come back to the situation where there is only one dependence constraint, no other constraint of form (2) having the possibility to be generated in this case.

Definition 6.5.3. *The second complexity measur*e of a parametrized T, we denote by $mes_2(T)$, is the maximum length for the set of all the constraints boxes, among the lengths of the terms v occurring in the constraints of form (2), namely expressed as $f_i = v(f_j) + m$.

Now we show the invariance of these two kinds of measure by the process of extending trees.

Lemma 6.5.1. *The first and second complexity measures of a tree extending an original tree T are, respectively, equal to the corresponding measures of T.*

Proof. By analysis of cases occurring in the definition of an extended tree. $\square$

Proof of Proposition 6.5.1. Let T be a given parametrized tree. According to Section 6.4 one can be satisfied just to consider trees which are α-β-1-reduced. Let p be the first complexity measure of T. Assume $p \neq 0$. For each constraints box B_i of T, there are perhaps several constraints of the form (2) as follows: $f_i = v(f_j) + m$, with v having length p. For a given box, we consider the set E_i of constraints of the form $f_i = v(f_j) + m$, with v having length p. If E_i is not empty, then we call *reference constraints* of this box the unique constraint belonging to E_i such that firstly j is minimum (within the members of E_i), secondly v is minimum for the lexicographic ordering on $\{K, L\}^*$ and, finally, m is the smallest possible integer.

Let us denote by C the set of constraints of form (2) as $f_i = u(f_k) + s$ of T which are not equal to some reference constraints, for the i's such that the corresponding E_i

is not empty. Let us denote by T' the tree we obtain from T by removing from the constraints boxes all constraints belonging to C. The first complexity measure of T' is strictly greater than p. Let us extend T', supposed to be of height m, into trees having as common height $m+p$. Each one of these extended tree must be treated as follows.

Let T'' be anyone of these extended trees. From Lemma 6.5.1, T'' has the same first complexity measure as T'. Since we did not take in account the constraints of C, we do not have at the moment an equivalent (to T) family of trees. In order to restore this equivalence, we shall deal one by one with the constraints of C.

Case 1: If C contains the constraint $t(f_i)=q$, where i has a non-zero length, then $t(v(f_j)+m)=q$ holds so that, by normalization, certain constraints of form (1) and (1′) must be placed within B_j.

We deal with Case (1′) in a similar way.

Case 2: Let $f_i=u(f_k)+s$ be a constraint belonging to C. Let $f_i=v(f_j)+m$ be the reference constraint of the box B_i. The ordered pair of constraints $\langle f_i=v(f_j)+m,\ f_i=u(f_k)+s\rangle$ is equivalent to the ordered pair of constraints $\langle f_i=v(f_j+m,\ v(f_j)=u(f_k)+s-m\rangle$. Since the length of u is greater or equal to p, the term u can be written as $u=u'u''$, with u'' having length p. The terms $v(f_j)$ and $u''(f_k)$ are leaves of the tree T'', say $f_{m+p,a}(x)$ and $f_{m+p,b}(x)$. So doing, we shall get a family of trees T''' equivalent to the original T after having for any considered ordered pair of such constraints and for any extended tree T'':

- adding in the box B_a the constraint $f_a=u'(f_b)+s-m$ if $f_a\neq f'_b$;
- cloning this tree on finitely many ones by adding for each one of them in the box B_a a constraint of the form $f_a=constant$ when $a=b$, the set of constants being according to Section 6.3, the finite set of the solutions of the equation $u'(f)=f+m-s$.

The obtained trees T''' have not necessarily a first complexity measure equal to 0 since, on the one hand, there are reference constraints $f_i=v(f_j)+m$ where v has a length strictly greater than p and, on the other, there are may be new constraints of the form (2) as $f_a=u'(f_b)+s-m$ we have possibly re-introduced during the transformation of T'' into T'''. However we can prove the following fact.

During the previous step,

either $mes_1(T''')=0$,

or $mes_1(T''') < mes_1(T)$,

or $mes_2(T''') < mes_2(T)$ holds.

Proof of the fact within the proof of Proposition 6.5.1. The first complexity measure of T' is p since we just keep, among the constraints of the form (2), the reference constraint. Therefore the first complexity measure of T'' is, according to the fact, equal to p, with $p\geqslant 1$. For a constraint $f_a=u'(f_b)+s-m$ we introduce the length of the term u' verifying

$$lg(u') = lg(u) - lg(u'') = lg(u) - p \leqslant lg(u) - 1.$$

Let us distinguish three cases according to the fact that either the maximum of the $lg(u')$'s is zero, or strictly less than p (but different from zero) or greater than p. In the first case, the first complexity measure is zero and the proof is achieved. In the second case, the first complexity measure strictly decreases. In the third case, the second complexity measure has strictly decreased.

We are going back to the proof of Proposition 6.5.1. Beginning again, as many times as necessary, we obtain a tree having a first complexity measure which is null. The new constraints of the form $f_a = constant$ allow us to determine some more leaves.

Case 2′: If C contains the constraint $t(f_i) \neq w(f_k) + r$ then $t(v(f_j) + m) \neq w(f_k) + r$ holds. We are led to add certain constraints of forms (1) and (1′) and a constraint of the form (2′) within B_j.

Case 3′: if C contains the constraint $t(f_i) \neq v(y) + r$ then $t(v(f_j) + m) \neq v(y) + r$ holds. We are led to add certain constraints of forms (1) and (1′) and a constraint of type (3′) within B_j. □

Conclusion: After α-β-2-reduction, any constraints box contains
– either nothing,
– or a determining constraint,
– or a determining dependence constraint, say $f_i = v(f_j) + m$,
– or constraints of the forms (1) with t of non-zero length, or (1′) or (2′) or (3′), that is to say only determining constraints.

7. Consistency of parametrized trees

At the end of the simplification process, we find again a finite set G of ordered pairs, each one consisting of a simple parametrized tree T_i and of a formula $CONDEXT(T_i)$ of external conditions, together with the initial formula $CONDEXTOR$ of the starting point. In order to eliminate the existential quantification on x it suffices to take the disjunction of the external conditions $CONDEXT(T_i)$ such that the associated tree T_i is consistent, among the finite set of trees which are the first components of the ordered pairs of G. If there is no couple in F whose the first component is consistent, then the original formula is false. Actually, the decidability of $TH(\mathbb{N}, C)$ is reduced to the decision of the consistency of a simple tree, what we intend to do below.

7.1. An ordering of the leaves of a simple tree

Lemma 7.1.1. *Let T be a simple parametrized tree of length N. It does exist a permutation σ from $[0, 2^N - 1]$ such that*
(1) *If f_i is a determined leaf and f_j a non-determined leaf, then $\sigma(i) < \sigma(j)$.*
(2) *If the condition $f_j = v(f_i) + m$ lies in the box B_j, then $\sigma(i) < \sigma(j)$.*

Proof. It suffices to sort the leaves in the following way:
– firstly one places all determined leaves;

– then one places all the leaves f_i whose constraints boxes only contain fundamental constraints;
– finally one places the leaves f_i whose constraints boxes only contains the dependence constraint (which is therefore the only one belonging to B_j) $f_j = v(f_i) + m$ with f_j which is an already placed leaf.
– one iterates this last step up to the vanishment of all the leaves.

One can ask whether a problem can arise due to the appearance of a cycle, for instance

$$f_j = v(f_i) + m \quad \text{and} \quad f_i = w(f_j) + r.$$

But this is impossible owing to the saturation step otherwise we would get $f_i = u(f_i) + s$ and, according to Section 3, the leaf f_i would be determined. $\square$

7.2. Decidability of the consistency of the simple trees

The aim of the present section is to establish the following.

Proposition 7.2.1. *The consistency of the simple trees is decidable.*

Proof. We begin by performing some treatments to the given tree T. Let σ be one of the permutations whose existence is insured by Lemma 7.1.1 above.

First treatment: Re-organization of the fundamental constraints. Let f_i be a leaf of this tree whose constraints box contains only fundamental constaints, namely of the form
(1) $t(f_i) = m$ where t is of non-zero length;
(1′) $t(f_i) \neq m$;
(2′) $t(f_i) \neq t'(f_k) + m$ (with possibly $k = i$);
(3′) $t(f_i) \neq v(y) + m$.

Let us consider the constraints of form (2′).

For $k = i$, according to Section 3, one can replace such a constraint by finitely many constraints of form (1′).

For $\sigma(k) > \sigma(i)$, one must distinguish between three subcases. If constraints boxes of f_k only contain fundamental constraints then one can put these constraints in the box B_k and so we obtain an equivalent tree. If f_k is a determined leaf then one replaces this constraint by another one of form (1′) or (3′). If f_k is a dependent leaf, then the only constraint belonging to B_k is of the form $f_k = v(f_j) + r$ while the box B_j contains only fundamental constraints; if $j = i$ then we get finitely many constraints of form (1′) which are equivalent; otherwise we get a constraint (2′) which is equivalent and, depending of f_i and f_j, we put it either in B_i or B_j according to $\sigma(i) > \sigma(j)$ or not. After this first treatment, one can assume that, for the constraints of form (2′), the inequality $\sigma(k) < \sigma(i)$ holds.

Second treatment: Regrouping the determining constraints. The constraints of form (1) as $t(f_i) = m$ where t has a non-zero length, which are in boxes just containing fundamental constraints are cumbersome for deciding consistency. One cannot completely eliminate them, but it is interesting to reduce them to something convenient. If we have simultaneously in a constraints box, $Ku(f_i) = a$ and $Lu(f_i) = b$ then we replace these two constraints by the equality $u(f_i) = C(a,b)$. One iterates this process while there is no more possible such reductions.

Third treatment: Elimination of certain disequations. Let us consider an ordered pair of constraints of the respective forms $t(f_i) = m$ and $u(f_i) \neq v(f_j) + r$, where t is a suffix of u. Then we have $u = u'.t$. Therefore the second constraint is equivalent to $u'(m) \neq v(fj) + r$, or in other words to $v(f_j) \neq u'(m) - r$. Now it suffices to remove the second constraint from the box B_i and to put the new one in B_j.

However by this way the obtained tree is not necessarily simple. If the box B_j did not contain any constraint, it is afterwards of type (d) of the definition of simple trees. If this box was determined, then we see immediately whether we introduce a contradiction (so that the tree is inconsistent) or if the condition is redundant. If the considered box contained only fundamental constraints then we just add one more condition. If the box contained the unique dependence constraint $f_j = w(f_k) + s$ then the leaf f_k is necessarily of type (d).

One can replace $f_j = w(f_k) + s$ by $v(w(f_k) + s) \neq u'(m) - r$. By normalization of this latter formula, we obtain an equivalent family of trees by adding within each tree only constraints of forms (1) and (1′) within B_k, so that the obtained trees are simple.

Let us consider an ordered pair of constraints of the respective forms $t(f_i) = m$ and $u(f_i) \neq v(y) + r$, where t is a suffix of u. Then we have $u = u'.t$. Therefore the second constraint is equivalent to $u'(m) \neq v(y) + r$, or in other words to $v(y) \neq u'(m) - r$. Now it suffices to remove the second constraint from the box B_i and to put the new one in $CONDEXT(T)$.

After this treatment when we have an ordered pair of constraints, say ($t(f_i) = m$, $u(f_i) \neq v(f_j) + r$) in any constraints box, then t is not a suffix of u.

Determination of the consistency of simple trees after the previous treatments: Let $T(x)$ be a simple parametrized tree treated as above. We intend to apply to this tree a process allowing us either to construct an integer which satisfies it, or to conclude this tree is inconsistent. For this purpose, we inspect all the leaves from the leaf $f_{\sigma(0)}$ until the leaf $f_{\sigma(2^n-1)}$ following the linear order we have introduced above and assigning them values in order to compute the value of the root x. We shall deal successively with the following exhaustive cases.

Case 1: The constraints box of the leaf is empty. We assign to it an arbitrary value, say 0.

Case 2: The constraints box of the leaf contains a determining constraint $f_i = m$. We assign this value m to the leaf.

Case 3: The constraints box of the leaf f_i contains a dependence constraint $f_i = v(f_j) + r$, with $\sigma(j) < \sigma(i)$. The leaf f_j had previously received a value so that it is easy to compute the value we have to assign to the leaf f_i.

Case 4: The constraints box of the leaf f_i contains disequations, each one being of one of the following forms:
– $t(f_i) \neq m$;
– $t(f_i) \neq u(f_j) + m$, with $\sigma(j) < \sigma(i)$;
– $t(f_i) \neq v(y) + m$
and eventually also determining constraints $u(f_i) = r$, where u cannot be a suffix of a term t occurring in the disequations $t(f_i) \neq u(f_j) + m$ or $t(f_i) \neq v(y) + m$. Let us begin by considering only the constraints $u(f_i) = r$ and $t(f_i) \neq m$. It is easy to observe whether they are compatible. If they are not, then the tree is inconsistent. If they are compatible, we are going to show we can assign a value to such a leaf by choosing for the leaves f_j which occur within these constraints the previously choosen values (since $\sigma(j) < \sigma(i)$). Let a' be the maximum of the values m, $v(y) + m$ and $u(f_j) + m$ which occur in the disequations. Put $a = a' + 1$. Let p be the maximum of length among the terms t and u occurring in the above constraints. One considers the tree having p as its height and f_i as its root. One enumerates the leaves of this auxiliary tree in a natural way from left to right. If a node $u(f_i)$ is determined, that is to say if there exists a constraint of the form $u(f_i) = r$, then it is the same (via a new saturation) for its descending nodes and, in particular, of the leaves of the subtree having $u(f_i)$ as its root. One gives the adequate value to the determined leaves and the value a to all non-determined leaves, what attibutes some value to f_i. Now we prove the leaf f_i verifies all the needed constraints.

Owing to the assigned values, the determining constraints are satisfied. Next, let us consider a disequational constraint which bears on $t(f_i)$. If all the leaves of the subtree having $t(f_i)$ as a root would be determined, then it would be the same situation for $t(f_i)$. Since u is the suffix of no t, this is impossible. Consequently, there is at least a leaf which had received the value a. From $x > K(x)$ and $x > L(x)$ for $x > 1$, we deduce $t(f_i) \geqslant a$, so that the considered disequation is satisfied by x. $\square$

Therefore, the initial original formula is equivalent to a disjunction of $CONDEXTOR$ and of formulas $CONDEXT(T_i)$. So, the existential quantifier bearing on x is eliminated.

The final formula, i.e. obtained after elimination of all quantifiers, will be a boolean combination of atoms of the form $m = n$, where m and n are explicitely determined integers. One easily can determine the truth value of such a formula. So we do have a decision algorithm for the sentences ot the structure $\langle \mathbb{N}, C, S \rangle$.

References

[1] G. Cantor, R. Dedekind, Briefwelrel, Hermann, Paris, 1937; traduction française in: J. Cavaillès, Philosophie mathématique, Hermann, Paris, 1962, pp. 177–249.

[2] G. Cantor, Uber eine Eigenschaft des Inbegroffes aller reellen algebraischen Zahlen, J. Reine Angew. Math. 77 (1874) 258–262; = Gesamm. abh., pp. 15–118, Springer, Berlin, 1930; traduction française in Acta Math. 2 (1883) 305–310.

[3] P. Cegielski, Definability, decidability and complexity, Ann. Math. Artif. Intell. 16 (1996) 311–341.

[4] P. Cegielski, D. Richard, On arithmetical first-order theories allowing encoding and decoding of lists, Theoret. Comput. Sci. 222 (1999) 55–75.
[5] K.J. Compton, C.W. Henson, A uniform method for proving lower bounds on the computational complexity of logical theories, Ann. Pure Appl. Logic 48 (1990) 1–79.
[6] H.B. Enderton, A Mathematical Introduction to Logic, Academic Press, New York, 1972, XIII + 295 p.
[7] J. Ferrante, C. Rackoff, The Computational Complexity of Logical Theories, Lecture Notes in Mathematics, vol. 718, Springer, Berlin, 1979, X + 243 p.
[8] I. Korec, A list of arithmetical structures complete with respect to the first-order definability, Theoret. Comput. Sci. 257 (this Vol.) (2001) 115–151.
[9] J. Robinson, Definability and decision problems in arithmetic, J. Symbolic Logic 14 (1949) 98–114; rep. in: S. Feferman (Ed.), The Collected Works of Julia Robinson, American Mathematical Society, Providence, RI, 1996, 338. p.
[10] R. Tenney, Decidable pairing functions, manuscript, 1974, Department of Computer Science, Cornell University.
[11] P. Cegielski, S. Grigorieff, D. Richard, La théorie élémentaire de la fonction de couplage de Cantor des entiers naturels est décidable, Comptes rendus de l'Académie des Sciences, Paris, to appear.