REPORTS ON MATHEMATICAL LOGIC
46 (2011), 91–108

Tomasz A. GORAZD and Jacek KRZACZKOWSKI

THE COMPLEXITY OF PROBLEMS CONNECTED WITH TWO-ELEMENT ALGEBRAS

A b s t r a c t. This paper presents a complete classification of the complexity of the SAT and equivalence problems for two-element algebras. Cases of terms and polynomials are considered. We show that for any fixed two-element algebra the considered SAT problems are either in P or NP-complete and the equivalence problems are either in P or coNP-complete. We show that the complexity of the considered problems, parametrized by an algebra, are determined by the clone of term operations of the algebra and does not depend on generating functions for the clone.

1. Introduction

One of the oldest and best known problems on the border between mathematics and computer science is to decide whether an equation has a so-

Received 25 September 2009

lution. From ancient times mathematicians studied equations of various forms over integers, real and complex numbers. We call this type of problems *satisfiability problems*. One of the first and most well known results in complexity theory is the NP completeness of the SAT problem - the satisfiability of Boolean formulas in CNF form. In computer science the interest in the equation satisfiability problem for finite algebraic structures has been increasing in recent years. The majority of the papers consider equations and systems of equations between terms or polynomials over a finite algebra with a fixed language. There are results concerning groups, monoids, semigroups, rings or lattices (see [7], [2],[9] or [17]). In [13] Larose and Zádori consider the complexity of a system of polynomial equations over arbitrary algebras and give, among others, the complete solution for algebras in congruence modular varieties.

The *term (polynomial) equivalence problem* asks if two given terms (polynomials) define the same function over a fixed algebra. There are many complexity results for this problem for finite monoids and semigroups [12], rings [10, 4] and groups [5, 9, 8].

For a fixed algebra the satisfiability problems are in the complexity class NP and the equivalence problems in the class coNP. One can ask if for any algebra the considered problem is always in P or NP-complete (P or coNP-complete)? For example, the problem of the satisfiability of a system of polynomial equations over a group $\mathbf{G}$ is in P if $\mathbf{G}$ is abelian and NP-complete otherwise ([7, 13]).

One of the most widely known subclasses of NP which exhibits such a dichotomy, is the class of constraint satisfaction problems (CSP) on the set $\{0, 1\}$, see [16]. Recently Bulatov proved the dichotomy for CSP on a three-element set [3].

In this paper we consider two-element algebras. We give a full classification of the term (polynomial) solvability and term (polynomial) equivalence problems for these algebras. We show the dichotomy for these problems. In the case of the satisfiability of a system of term (polynomial) equations apart from showing the dichotomy, which can also be deduced from [13] and [16], we show that for the NP-completeness we need only two equations. This can not be obtained using the methods from [13, 16].

In [9] the authors ask if there exists an algebra for which the polynomial equivalence problem is hard and the polynomial satisfiability problem is in P. We show infinitely many two-element algebras with this property; one

of them is the two-element lattice.

We also ask how the complexity of a problem depends on the presentation of an algebra. We show that in all considered cases the complexity is representation independent, i.e the complexity of the problem is equal for any two algebras that generate the same set of term functions.

2. Preliminaries

A *language* (or *type*) of algebras is a set $\mathcal{F}$ of function symbols with a nonnegative integer assigned to each member of $\mathcal{F}$. This integer is called the arity of $f \in \mathcal{F}$.

An *algebra* of type $\mathcal{F}$ is an ordered pair $\mathbf{A} = (A, F)$ where A is a nonempty set (called *universe*) and F is a family of finitary operations on A indexed by the language $\mathcal{F}$ such that for any n-ary symbol $f \in \mathcal{F}$ there is an n-ary operation $f^{\mathbf{A}}$ on A. The $f^{\mathbf{A}}$'s are called *fundamental operations of* $\mathbf{A}$. If $F = \{f_1, \ldots, f_n\}$ it is customary to write $(A, f_1, \ldots, f_n)$ rather than (A, F). The subset of n-ary function symbols in F is denoted by F_n. We consider only algebras with 2 elements. Notice that the set of fundamental operations does not have to be finite.

For a language $\mathcal{F}$ and a set of variables X ($|X| = \omega$) we define $T(X)$, the set of terms of type $\mathcal{F}$, as the smallest set such that

- $X \cup \mathcal{F}_0 \subseteq T(X)$
- If $t_1, \ldots, t_n \in T(X)$ and $f \in \mathcal{F}_n$, then the "string" $f(t_1, \ldots, t_n) \in T(X)$

If for an algebra $\mathbf{A}$ of type $\mathcal{F}$ we additionally admit all constant operation symbols on A while building terms, we get the *polynomials* of $\mathbf{A}$.

If $\mathbf{A}$ is an algebra of type $\mathcal{F}$, then with terms and polynomials we can associate operations on the set A in an obvious way. If t is a term (polynomial) in which only the (distinct) variables from $\{x_1, \ldots, x_n\}$ appear, then $t^{\mathbf{A}}(x_1, \ldots, x_n)$ describes the corresponding n-ary term (polynomial) operation. The set of term operations on $\mathbf{A}$ we denote by $Clo(\mathbf{A})$ and the set of polynomial operations we denote by $Pol(\mathbf{A})$. Observe that these sets are *clones of operations* on A, i.e. sets of operations on A, closed under composition, and containing the projection operations $\pi_i^n(x_1, \ldots, x_n) = x_i$.

The variety generated by an algebra $\mathbf{A}$ of the type $\mathcal{F}$ is the smallest class of algebras of the type $\mathcal{F}$ containing $\mathbf{A}$ and closed under subalgebras, homomorphic images and direct products. We denote such a variety $V(\mathbf{A})$. If two algebras generate the same variety then these algebras have the same identities. For more details see [6].

In this paper the main method for proving NP-completeness will be the rewriting of terms from one language to another. Let t be a term of the form $s(t_1, \ldots, t_k)$ where s is a function symbol and $t_1, \ldots, t_k$ are terms. Let w be a term with variables $x_1, \ldots, x_l$, $l \geq k$. A *substitution* of the function symbol s in the term t by the term w is the term constructed from w by substituting every occurrence of x_i by the string $(t_i), 0 \leq i \leq k$.

Because we are interested in problems over two-element algebras we will make essential use of the Post classification for clones on the two-element set. The Hasse diagram of the order on the set of such clones is presented in the following figure. We use the original Post notation for clones on the set $2 = \{0, 1\}$, however, we recall them here for the reader's convenience.

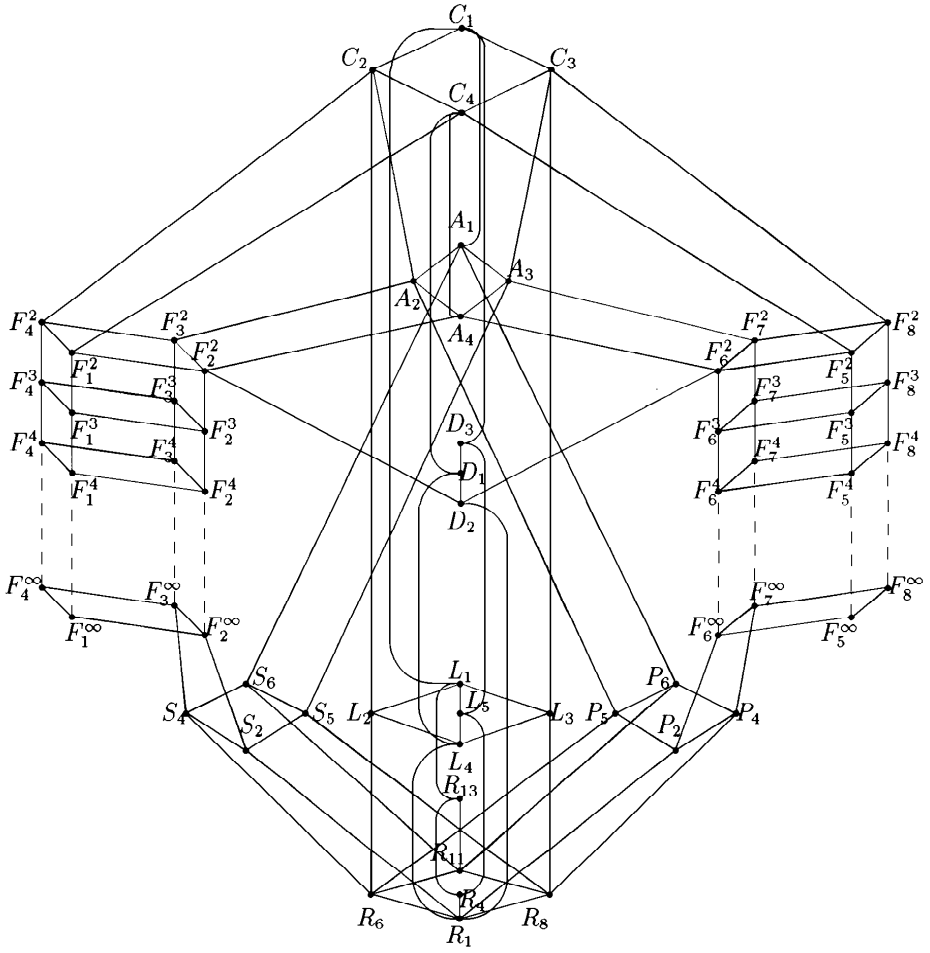
$\mathbf{C_1} = (2, \wedge, \vee, \neg)$	$\mathbf{C_3} = (2, -, \vee)$	$\mathbf{C_4} = (2, \vee, ki)$
$\mathbf{A_1} = (2, \wedge, \vee, 0, 1)$	$\mathbf{A_3} = (2, \wedge, \vee, 0)$	$\mathbf{A_4} = (2, \wedge, \vee)$
$\mathbf{D_3} = (2, d, \neg)$	$\mathbf{D_1} = (2, d, +_3)$	$\mathbf{D_2} = (2, d)$
$\mathbf{L_1} = (2, +, \neg)$	$\mathbf{L_3} = (2, +)$	
$\mathbf{L_5} = (2, +_3, \neg)$	$\mathbf{L_4} = (2, +_3)$	
$\mathbf{F_8^m} = (2, -, d_m)$	$\mathbf{F_8^\infty} = (2, -)$	
$\mathbf{F_7^m} = (2, ka, d_m, 0)$	$\mathbf{F_7^\infty} = (2, ka, 0)$	
$\mathbf{F_6^m} = (2, ka, d_m)$	$\mathbf{F_6^\infty} = (2, ka)$	
$\mathbf{F_5^m} = (2, ki, d_m)$	$\mathbf{F_5^\infty} = (2, ki)$	
$\mathbf{P_6} = (2, \wedge, 0, 1)$	$\mathbf{P_5} = (2, \wedge, 1)$	
$\mathbf{P_4} = (2, \wedge, 0)$	$\mathbf{P_2} = (2, \wedge)$	
$\mathbf{R_{13}} = (2, \neg, 0)$	$\mathbf{R_4} = (2, \neg)$	
$\mathbf{R_{11}} = (2, 0, 1)$	$\mathbf{R_8} = (2, 0)$	
$\mathbf{R_1} = (2)$		

where

$x - y = x \wedge \neg y$,
$ki(x, y, z) = x \wedge (y \to z)$,
$ka(x, y, z) = x \wedge (y \vee z)$,
$+_3(x, y, z) = x + y + z \ \ (\ mod\ 2)$,
$d_m(x_0, \ldots, x_m) = \bigvee_{i=0}^{m}(x_0 \wedge \ldots \wedge x_{i-1} \wedge x_{i+1} \wedge \ldots \wedge x_m), \quad m \geq 2,$

$d = d_2$.

These describe the central and right-hand side parts of the diagram.



The clones $C_2, A_2, L_2, R_6, F_i^\alpha$ and S_i are dual to $C_3, A_3, L_3, R_8, F_{i+4}^\alpha$ and P_i, respectively, in the sense that an operation $\mathbf{f}$ is dual to $\mathbf{g}$ if $\mathbf{f}(x_1, \ldots x_k) = \neg\mathbf{g}(\neg x_1, \ldots, \neg x_k)$, i.e. the clone X is dual to Y if the map $\neg : 2 \longrightarrow 2$ is an isomorphism of $(2, X)$ onto $(2, Y)$.

A function $f : 2^n \to 2$ is called

- monotone iff

$$(\forall_{0 \le i < n}\ a_i \le b_i) \Rightarrow f(a_0, \ldots, a_{n-1}) \le f(b_0, \ldots, b_{n-1}),$$

- 0-valid iff $f(0,\dots,0)=0$,
- 1-valid iff $f(1,\dots,1)=1$.

The main question considered in this paper is the complexity of the following problem:

Definition 2.1. *For an algebra* $\mathbf{A}$ *the term satisfiability problem (*TERM-SAT($\mathbf{A}$)*) is the decision problem with*

Instance: *A pair of terms* (s,t) *with the tables of the fundamental operations of* $\mathbf{A}$ *corresponding to all function symbols occurring in* s *and* t.

Question: *Does a substitution of variables from* s *and* t *by values from* A *exist such that the values of the functions* $s^{\mathbf{A}}$ *and* $t^{\mathbf{A}}$ *are the same?*

The tables of the fundamental operations occuring in the instance of TERM-SAT($\mathbf{A}$) and presented in this instance make it possible to consider algebras with an infinite number of basic operations.

When in Definition 2.1 we replace terms by polynomials, we obtain the polynomial satisfiability problem (POL-SAT($\mathbf{A}$)).

We will also consider a set of equations instead of one equation from the previous definitions. In this case we will get SYS-TERM (SYS-POL), the problem of satisfiability of a system of equations between terms (polynomials).

The term (polynomial) equivalence problem for an algebra is the problem of deciding whether two terms (polynomials) define the same function. We denote these problems TERM-EQ and POL-EQ, respectively.

When describing instances of satisfiability and equivalence problems we often use $t_1 = t_2$ and $t_1 \approx t_2$, respectively. For the complement of equivalence problems we use $t_1 \not\approx t_2$.

In this paper we also ask whether the complexity of the considered problems depends on the representation of an algebra, i.e. if it is the same for any two algebras with equal termal clones? In general the answer is negative.

Example 2.2. *Consider the smallest non-nilpotent, solvable group* $\mathbf{S_3} = (S_3,\circ)$. *Let* $s(x,y,z,w) = x\circ[[[x,y],z],w]^{-1}$, *where* $[x,y] = x^{-1}\circ y^{-1}\circ x\circ y$. *Obviously* $Clo(S_3,\circ) = Clo(S_3,\circ,s)$.

POL-SAT($S_3, \circ$) *is in P ([9]) but* POL-SAT($S_3, s, \circ$) *is NP-complete ([11]).*

In order to distinguish cases where the complexity of a problem depends on the clone of term operations of an algebra, we introduce the following:

Definition 2.3. *Consider any decision problem over algebras. Let C be a clone. We call this problem*

- ***representation-independent*** *for C iff for any algebras $\mathbf{A}, \mathbf{B}$ such that $Clo(\mathbf{A}) = Clo(\mathbf{B}) = C$ the problem for $\mathbf{A}$ and the problem for $\mathbf{B}$ are polynomial-time equivalent.*
- ***representation-dependent*** *for C, otherwise.*

We say that a problem for a clone C is NP-complete (in P, coNP-complete) if it is representation-independent for C, and for every algebra $\mathbf{A}$ with $Clo(\mathbf{A}) = C$ the problem for $\mathbf{A}$ is NP-complete (in P, coNP-complete).

In this paper we will prove that for two-element algebras the problems TERM-SAT, POL-SAT, SYS-TERM and SYS-POL are representation independent, moreover they are either NP-complete or in P. Also TERM-EQ and POL-EQ are representation independent and coNP-complete or in P.

Notice that for a term (polynomial) we can compute the value of the corresponding function (for given arguments) in polynomial time. Consequently all problems considered in this paper are in NP (satisfiability problems) or coNP (equivalence problems).

3. Satisfiablility of an equation

Let us start with the TERM-SAT for primal algebras, i.e. algebras with the termal clone equal to C_1. In these algebras any function can be generated as a term function.

Lemma 3.1. TERM-SAT *for primal algebras is NP-complete.*

Proof. Let $\mathbf{A}$ be a primal algebra. We will polynomially encode 3-SAT in TERM-SAT($\mathbf{A}$).

Notice that in the language of $\mathbf{A}$ we can generate the operations $\wedge$, $\vee$ and $\neg$ as term functions. We call the corresponding terms $t_\wedge(x, y)$, $t_\vee(x, y)$

and $t_\neg(x)$, respectively. We can also generate the constant 1 as a term function.

Let

$$C_1 \wedge C_2 \wedge \ldots \wedge C_n \tag{1}$$

be an instance of 3-SAT, $C_i = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$ and every x_{i_j} is a variable or a negated variable.

Observe that in $t_\wedge(x, y)$ we can have multiple occurrences of x or y. Hence the replacement of $\wedge$ in (1) by $t_\wedge(x, y)$ step by step from the left to the right could produce an expression exponentially longer then the input. To avoid such a situation we use the divide and conquer paradigm in the following algorithm.

Encode(S)
Input: 3-SAT instance S of the form (1).
Output: A term in the language of $\mathbf{A}$ whose term function is equal to S (identify 0 with $false$ and 1 with $true$).
if *S is of the form C_1* **then**
  return C_1 after replacing $\vee$ and $\neg$ by $t_\vee(x, y)$ and $t_\neg(x)$, respectively
else
  let $S = S_1 \wedge S_2$ and the numbers of occurrences of $\wedge$ in S_1 and S_2, respectively, differ by at most 1.
  return $t_\wedge(Encode(S_1), Encode(S_2))$
end

Algorithm 1: *Encode*(S)

Algorithm 1 works in polynomial time. Notice that the depth of the recursion in the above algorithm is logarithmic in the size of the input. Hence because $|t_\wedge(t_1, t_2)| = O(|t_1|+|t_2|)$ the size of the output is polynomial in the size of the input.

Now for a 3-SAT instance S the corresponding TERM-SAT($\mathbf{A}$) instance is the following:

$$Encode(S) = 1 \tag{2}$$

One can see that S is satisfiable iff (2) is. □

There is only one more clone where TERM-SAT in NP-complete. This is $Clo(\mathbf{D_3})$.

Lemma 3.2. TERM-SAT *for the clone* $Clo(\mathbf{D_3})$ *is NP-complete.*

Proof. Let $\mathbf{A}$ be an algebra with $Clo(\mathbf{A}) = Clo(\mathbf{D_3})$. If to the basic operations of $\mathbf{A}$ we add the constant operation 1 we obtain a primal algebra. Denote the new algebra by $\mathbf{A}'$.

Call $\neg$ a term in the language of $\mathbf{A}$ representing negation.

We will define a reduction from 3-SAT to TERM-SAT($\mathbf{A}$). Let S be a 3-SAT instance. First we verify whether S is a tautology. This can be easily done in a polynomial time. Every tautology we reduce to the equation $x = x$.

If S is not a tautology we run $Encode(S)$ (Algorithm 1) over $\mathbf{A}'$ and we obtain a term $t'(x_1, \ldots, x_k)$. Next in $t'(x_1, \ldots, x_k)$ we replace every occurrence of the constant 1 by a new variable u. We call the new term $t(x_1, \ldots, x_k, u)$. Now we reduce S to the following instance of TERM-SAT($\mathbf{A}$)

$$t(x_1, \ldots, x_k, u) = t(x'_1, \ldots, x'_k, \neg u), \tag{3}$$

where $\{x_1, \ldots, x_k\} \cap \{x'_1, \ldots, x'_k\} = \emptyset$.

We have to show that the procedure described above is in fact a reduction.

One can see that all the above operations can be done in polynomial time and that the size of $t(x_1, \ldots, x_k, u)$ is at most polynomially larger then the size of S.

Observe that for every term g in the language of $\mathbf{A}$ we have

$$\neg g(x_1, \ldots, x_m) = g(\neg x_1, \ldots, \neg x_m) \tag{4}$$

If S is not satisfiable then $\forall_{(a_1,\ldots,a_k)\in 2^k}\ t^{\mathbf{A}}(a_1, \ldots, a_k, 1) = 0$ and therefore from (4) we have that $\forall_{(a_1,\ldots,a_k)\in 2^k}\ t^{\mathbf{A}}(a_1, \ldots, a_k, 0) = 1$. Consequently, (3) cannot be satisfiable.

Conversely, assume that S is satisfiable and not a tautology.

There are $(a_1, \ldots, a_k), (b_1, \ldots, b_k) \in 2^k$ such that $t^{\mathbf{A}}(a_1, \ldots, a_k, 1) = 0$ and $t^{\mathbf{A}}(b_1, \ldots, b_k, 1) = 1$. Now, using (4), we have that

$$t^{\mathbf{A}}(a_1, \ldots, a_k, 1) = t^{\mathbf{A}}(\neg b_1, \ldots, \neg b_k, 0)$$

Therfore (3) is satisfiable. □

We are now in a position to prove the main result of this section.

Theorem 3.3. TERM-SAT *for two-element clones is representation-independent. Moreover, it is NP-complete for* $Clo(\mathbf{C_1})$ *and* $Clo(\mathbf{D_3})$ *and is in P otherwise.*

Proof. The NP-complete part comes from Lemma 3.1 and Lemma 3.2. It remains to consider the following 4 classes of clones.

1. 1-valid, i.e. clones contained in $Clo(\mathbf{C_2})$,
2. 0-valid, i.e. clones contained in $Clo(\mathbf{C_3})$,
3. monotone, i.e. clones contained in $Clo(\mathbf{A_1})$,
4. affine, i.e clones contained in $Clo(\mathbf{L_1})$.

Term equations in the first two classes are always satisfiable. In the third case it is enough to consider only the values of the term functions for $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$. The affine functions are all the functions we can obtain as polynomial functions over $\mathbf{L_3}$. In this case an equation $t_1 = t_2$ has a solution iff the equation $t_1 + t_2 = 0$ also has a solution. The second equation does not have any solution iff the term on the left-hand side defines the constant function 1. To recognize such a situation observe that an affine function $f(x_1, \ldots, x_k)$ depends on the variable x_i iff $f(0, 0, \ldots, 0) \neq f(0, 0, \ldots, 1, \ldots, 0)$, where 1 is at the ith position. This concludes the proof. □

Having the characterization of the complexity of TERM-SAT, we can easily prove the following corollary which describes the complexity of POL-SAT.

Corollary 3.4. POL-SAT *for two-element clones is representation-independent. Moreover, it is NP-complete for algebras where the polynomial clone consists all operations and is in P otherwise.*

Proof. Let $\mathbf{A}$ be a two element algebra. Define $\mathbf{A'}$ by adding the constant operations to $\mathbf{A}$. Observe that POL-SAT($\mathbf{A}$) is the same as TERM-SAT($\mathbf{A'}$). Therfore, by Theorem 3.3, POL-SAT is representation-independent.

Moreover $Pol(\mathbf{C_1}) = Pol(\mathbf{D_1}) = Pol(\mathbf{F_1^\infty}) = Pol(\mathbf{F_5^\infty})$ and consequently for clones including $Clo(\mathbf{D_1})$ or $Clo(\mathbf{F_1^\infty})$ or $Clo(\mathbf{F_5^\infty})$, POL-SAT is NP-complete.

It remains to consider clones contained in $Clo(\mathbf{A_1})$ or $Clo(\mathbf{L_1})$. In these cases POL-SAT is in P because $Pol(\mathbf{A_1}) = Clo(\mathbf{A_1})$ and $Pol(\mathbf{L_1}) = Clo(\mathbf{L_1})$, and therefore for solving POL-SAT we can use algorithms solving TERM-SAT for algebras with termal clones equal to $Clo(\mathbf{A_1})$ or $Clo(\mathbf{L_1})$. □

4. Satisfiability of a system of equations

In this section we consider the satisfiability of systems of equations between terms or between polynomials. Larose and Zádori in [13] show a characterization of the computational complexity of SYS-POL for algebras of a finite type in congruence modular varieties (Corollary 3.14). Using this information and the facts from the Schaefer paper [16] it is possible to characterize the complexity of SYS-POL for two-element algebras. In this paper we give a simple direct proof for SYS-TERM and SYS-POL for two-element algebras, and we show that for the NP-completeness we only need systems of two equations.

Theorem 4.1. SYS-TERM *for two-element algebras is representation-independent. For an algebra* $\mathbf{A}$ *the problem* SYS-TERM($\mathbf{A}$) *is NP-complete if*

- $Clo(\mathbf{A}) = Clo(2, \wedge, \vee, \neg) = Clo(\mathbf{C_1})$
- $Clo(\mathbf{A}) = Clo(2, d, \neg) = Clo(\mathbf{D_3})$
- $Clo(\mathbf{A}) = Clo(2, \wedge, \vee, 0, 1) = Clo(\mathbf{A_1})$

and is in P otherwise. Moreover for algebras where SYS-TERM *is NP-complete the problem of satisfiability of two equations is also NP-complete.*

Proof. The proof of the first two cases of the NP-complete part follows immediately from Theorem 3.3.

Let $Clo(\mathbf{A}) = Clo(2, \wedge, \vee, 0, 1)$. In the language of $\mathbf{A}$ we have terms defining the functions $0, 1, \wedge, \vee$. Denote these terms $t_0(x)$, $t_1(x)$, $t_\wedge(x, y)$, $t_\vee(x, y)$, respectively.

Consider a subproblem of 3-SAT where the instances are of the form

$$\bigwedge_{(x,y,z)\in Y} (x \vee y \vee z) \wedge \bigwedge_{(x,y,z)\in Z} (\neg x \vee \neg y \vee \neg z) = 1 \tag{5}$$

and Y, Z are finite sets of triples of variables. This problem is NP-complete; one can prove this using results from [16]. We will show a polynomial time reduction of this problem to SYS-TERM($\mathbf{A}$).

Obviously, (5) is satisfiable iff the system

$$\begin{cases} \bigwedge_{(x,y,z)\in Y}(x \vee y \vee z) &= 1 \\ \bigvee_{(x,y,z)\in Z}(x \wedge y \wedge z) &= 0 \end{cases} \tag{6}$$

has a solution.

Now in order to finish this part of the proof observe that we can construct terms in the language of $\mathbf{A}$ equivalent to the terms on the left hand side of (6) in polynomial time. We can do it using the procedure *Encode*(Algorithm 1) for the first term. For the second one use *Encode* with $\vee, t_\vee, \wedge$ and $t_\wedge$ substituted by $\wedge, t_\wedge, \vee$ and $t_\vee$, respectively.

For the polynomial part of the proof we have to consider algebras with termal clones that are subclones of $Clo(\mathbf{C_2})$, $Clo(\mathbf{C_3})$, $Clo(\mathbf{L_1})$, $Clo(\mathbf{P_6})$ or $Clo(\mathbf{S_6})$.

In the first two cases the systems are always satisfiable. In the third one we can express every term as +'s of variables or the constant 1 in polynomial time (see the proof of Theorem 3.3). Next use the Gaussian elimination in order to solve this system of equations.

Now let $\mathbf{A}$ be an algebra with $Clo(\mathbf{A}) \subseteq Clo(\mathbf{P_6})$. Let $t(x_1, \ldots, x_k)$ be a term in the language of $\mathbf{A}$. Observe that $t(x_1, \ldots, x_k)$ defines the constant 1 iff $t^{\mathbf{A}}(0, \ldots, 0) = 1$ and it defines 0 iff $t^{\mathbf{A}}(1, \ldots, 1) = 0$. If $t^{\mathbf{A}}(x_1, \ldots, x_k)$ is not a constant operation then $t^{\mathbf{A}}(x_1, \ldots, x_k) = x'_1 \wedge x'_2 \wedge \ldots \wedge x'_l$ where $\{x'_1, x'_2, \ldots, x'_l\} \subseteq \{x_1, \ldots, x_k\}$. One can see that $x_i \in \{x'_1, x'_2, \ldots, x'_l\}$ iff $t^{\mathbf{A}}(1, \ldots, 0 \ldots, 1) = 0$, where 0 is at the ith position. Using this information for a given system of equations we can construct an equivalent system where the terms are constants or conjunction of variables. It can be done in polynomial time and the size of the new system is at most polynomialy larger then size of the original one. Now one can see that the last system has a solution iff it is satisfiable by the valuation w of variables such that

$$w(x) = 1 \text{ iff } x \text{ occurs in an equation of the form } x_1 \wedge x_2 \wedge \ldots \wedge x_k = 1$$

The case of $Clo(\mathbf{S_6})$ is symmetric to $Clo(\mathbf{P_6})$. □

Corollary 4.2. SYS-POL *for two element algebras is representation-independent. For an algebra* $\mathbf{A}$ *the problem* SYS-POL($\mathbf{A}$) *is NP-complete*

if

$$Clo(\mathbf{F}_{\mathbf{6}}^{\infty}) \subseteq Clo(\mathbf{A}) \text{ or } Clo(\mathbf{F}_{\mathbf{2}}^{\infty}) \subseteq Clo(\mathbf{A}) \text{ or } Clo(\mathbf{D_2}) \subseteq Clo(\mathbf{A})$$

and is in P otherwise.

Moreover for algebras where SYS-POL *is NP-complete the problem of satisfiability of two equations is also NP-complete.*

Proof. Obviously SYS-POL for an algebra is the same as SYS-TERM for a new algebra arising from the previous one by adding the constant operations. Then the corollary is an immediate consequence of Theorem 4.1. □

For brevity, the above corollary can be restated as follows:

Corollary 4.3. *For an algebra* $\mathbf{A}$ *the problem* SYS-POL($\mathbf{A}$) *is NP-complete if the variety generated by* $\mathbf{A}$ *is congruence distributive and is in P otherwise.*

5. Term equivalence

The equivalence problems considered in this section lie in the coNP class. To show the coNP completeness of a problem we will use polynomial time reduction to encode in it a coNP-complete problem or we will encode an NP complete problem in the complement of the considered problem.

Lemma 5.1. *For an algebra* $\mathbf{A}$ *with* $Clo(\mathbf{A})$ *equal to* $Clo(\mathbf{C_1}), Clo(\mathbf{D_3})$ *or* $Clo(\mathbf{A_1})$ *the problem* TERM-EQ($\mathbf{A}$) *is coNP-complete.*

Proof. We use the symbols $\neg, \vee$ for terms with term functions equal to negation and disjunction, respectively. First observe that for an algebra $\mathbf{A}$ such that TERM-SAT for $Clo(\mathbf{A})$ is NP-complete and there is the negation in $Clo(\mathbf{A})$ then TERM-EQ for $Clo(\mathbf{A})$ is coNP-complete. This follows immediately from the fact that the instance

$$t_1 = t_2$$

of TERM-SAT($\mathbf{A}$) can be reduced to the following instance of the complement of TERM-EQ($\mathbf{A}$):

$$t_1 \not\approx \neg t_2$$

Therefore we have coNP-completeness of TERM-EQ for $Clo(\mathbf{C_1})$ and $Clo(\mathbf{D_3})$.

Let us now take a look at $Clo(\mathbf{A_1})$. From the proof of Theorem 4.1 we know that for an algebra $\mathbf{A}$ with $Clo(\mathbf{A}) = Clo(\mathbf{A_1})$ the subproblem of SYS-TERM($\mathbf{A}$) where the instances are of the form

$$\begin{cases} t_1 &= 1 \\ t_2 &= 0 \end{cases} \tag{7}$$

is NP-complete. Now, because we have disjunction in the clone we can reduce (7) to

$$t_1 \vee t_2 \not\approx t_2$$

which is an instance of the complement of TERM-EQ($\mathbf{A}$). Hence TERM-EQ for $Clo(\mathbf{A_1})$ is coNP-complete. □

From now on, for an algebra $\mathbf{A}$ we denote by $\mathbf{A'}$ the algebra obtained by adding the constant operations 0 and 1 to the operations of $\mathbf{A}$. In the proofs of the following two lemmas we use the symbols ka, d either as the functions defined below the Post diagram or as terms defining these functions in the considered algebras; the way we use them will be clear from the context.

Lemma 5.2. *For an algebra* $\mathbf{A}$ *with* $Clo(\mathbf{F_2^\infty}) \subseteq Clo(\mathbf{A})$ *or* $Clo(\mathbf{F_6^\infty}) \subseteq Clo(\mathbf{A})$ *the problem* TERM-EQ($\mathbf{A}$) *is coNP-complete.*

Proof. Let $\mathbf{A}$ be an algebra with $Clo(\mathbf{F_6^\infty}) \subseteq Clo(\mathbf{A})$. Observe that $Clo(\mathbf{A'}) = Clo(\mathbf{A_1})$ or $Clo(\mathbf{A'}) = Clo(\mathbf{C_1})$ and therefore TERM-EQ($\mathbf{A'}$) is coNP-complete. Now the instance

$$t_1 \approx t_2 \tag{8}$$

of TERM-EQ($\mathbf{A'}$) we reduce to the following instance of TERM-EQ($\mathbf{A}$)

$$ka(x, t_1', y) \approx ka(x, t_2', y), \tag{9}$$

where x, y are new variables and t_1', t_2' are obtained from t_1, t_2, respectively, by replacing all the occurrences of 1 by x and 0 by y. To show that it is a reduction observe that $ka \in Clo(\mathbf{A})$ and therefore (9) is an instance of TERM-EQ($\mathbf{A}$). Now if (9) is true then it is also true after replacing x and y by 1 and 0, respectively. Because $ka(1, t, 0) \approx t$ then (8) must be

true. Conversely, assume that (9) is not true. This is only possible when $x = 1$ and $y = 0$ and consequently (8) is not true. Hence TERM-EQ($\mathbf{A}$) is coNP-complete.

The proof for an algebra $\mathbf{A}$ with $Clo(\mathbf{F}_{\mathbf{2}}^{\infty}) \subseteq Clo(\mathbf{A})$ is symmetrical to the previous case. □

Lemma 5.3. TERM-EQ *for* $Clo(\mathbf{D_1})$ *and* $Clo(\mathbf{D_2})$ *is representation independent and coNP-complete.*

Proof. For an algebra $\mathbf{A}$ such that $Clo(\mathbf{A}) = Clo(\mathbf{D_2})$ or $Clo(\mathbf{A}) = Clo(\mathbf{D_1})$ observe that $Clo(\mathbf{A}') = Clo(\mathbf{C_1})$ or $Clo(\mathbf{A}') = Clo(\mathbf{A_1})$. Hence TERM-EQ($\mathbf{A}'$) is coNP-complete. The reduction of TERM-EQ($\mathbf{A}'$) to TERM-EQ($\mathbf{A}$) for an instance $t_1 \approx t_2$ returns $d(t_1', x, y) \approx d(t_2', x, y)$, where x, y are new variables and t_1', t_2' are obtained from t_1, t_2, respectively by replacing all the occurrences of 1 by x and 0 by y. One can see that this is in fact a reduction, and therefore TERM-EQ($\mathbf{A}$) is coNP-complete. □

Theorem 5.4. TERM-EQ *for two element algebras is representation-independent. For an algebra* $\mathbf{A}$ *the problem* TERM-EQ($\mathbf{A}$) *is coNP-complete if*

$$Clo(\mathbf{F}_{\mathbf{6}}^{\infty}) \subseteq Clo(\mathbf{A}) \text{ or } Clo(\mathbf{F}_{\mathbf{2}}^{\infty}) \subseteq Clo(\mathbf{A}) \text{ or } Clo(\mathbf{D_2}) \subseteq Clo(\mathbf{A})$$

and is in P otherwise.

Proof. The coNP-complete part is a consequence of Lemmas 5.1, 5.2 and 5.3.

For the polynomial part of the lemma we only need to show polynomial time algorithms for algebras with termal clones equal to $Clo(\mathbf{L_1}), Clo(\mathbf{P_6})$ or $Clo(\mathbf{S_6})$. In the first case every term function can be expressed as a sum (+) of variables and constants. Having equations between terms in this form one can easily check their equivalence. All the above operations can be done in polynomial time (see the proof of Theorem 3.3).

For the second case observe that we can express every term function as a conjunction of variables or constants (see the proof of Lemma 4.1). Next it is easy to check the equivalence. All these operations can be done in polynomial time.

The case of $Clo(\mathbf{S_6})$ is symmetrical to the case of $Clo(\mathbf{P_6})$. This concludes the proof. □

Corollary 5.5. POL-EQ *for two element algebras is representation-independent. For an algebra* $\mathbf{A}$ *the problem* POL-EQ($\mathbf{A}$) *is coNP-complete if*

$$Clo(\mathbf{F_6^\infty}) \subseteq Clo(\mathbf{A}) \text{ or } Clo(\mathbf{F_2^\infty}) \subseteq Clo(\mathbf{A}) \text{ or } Clo(\mathbf{D_2}) \subseteq Clo(\mathbf{A})$$

and is in P otherwise.

For brevity, the above lemma and corollary can be restated as follows:

Lemma 5.6. *For an algebra* $\mathbf{A}$ *the problems* TERM-EQ($\mathbf{A}$) *and* POL-EQ($\mathbf{A}$) *are coNP-complete if the variety generated by* $\mathbf{A}$ *is congruence distributive and is in P otherwise.*

If two algebras define the same variety then they must have the same sets of identities. We can formulate this as follows.

Fact 5.7. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be algebras of the type* $\mathcal{F}$*. If* $V(\mathbf{A}) = V(\mathbf{B})$ *then for terms* t_1, t_2 *of type* $\mathcal{F}$ *we have* $t_1 \approx t_2$ *in* $\mathbf{A}$ *iff* $t_1 \approx t_2$ *in* $\mathbf{B}$*.*

Therefore as an immediate consequence of the above lemma we have the following:

Corollary 5.8. *Let* $\mathbf{A}$ *be an algebra with 2 or more elements. If* $\mathbf{A}$ *generates the same variety as a 2-element algebra, then* TERM-EQ($\mathbf{A}$) *is coNP-complete if this variety is congruence distributive and is in P otherwise.*

In [9] (Problem 1.) the authors ask if there exists an algebra $\mathbf{A}$ such that POL-SAT($\mathbf{A}$) is in P and POL-EQ($\mathbf{A}$) is coNP-complete. From Corollary 3.4 and Corollary 5.5 we have the following:

Corollary 5.9. *For every algebra* $\mathbf{A}$ *such that* $Clo(\mathbf{F_6^\infty}) \subseteq Clo(\mathbf{A}) \subseteq Clo(\mathbf{A_1})$ *or* $Clo(\mathbf{F_2^\infty}) \subseteq Clo(\mathbf{A}) \subseteq Clo(\mathbf{A_1})$ POL-SAT($\mathbf{A}$) *is in P and* POL-EQ($\mathbf{A}$) *is coNP-complete.*

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, 1974.

[2] D. M. Barrington, P. McKenzie, C. Moore, P. Tesson, D. Thérien, *Equation satisfiability and program satisfiability for finite monoids*, Math. Found. Comp. Sci., (2000), 127-181.

[3] A.Bulatov, *A dichotomy theorem for constraints on a three-element set,* Proceedings of the 43rd Symposium on Foundations of Computer Science (2002), 649-658.

[4] S. Burris and J. Lawrence, *The equivalence problem for finite rings,* Journal of Symbolic Computation, 15 (1993), 67-71.

[5] S. Burris, J. Lawrence, *Results on the equivalence problem for finite groups*, Algebra Universalis, 52(4) (2004), 495-500.

[6] S. Burris, H.P. Sankappanavar, *A Course in Universal Algebra*, Springer-Verlag, 1981.

[7] M. Goldmann, A. Russel, *The Complexity of Solving Equations over Finite Groups* Inf. Comput., 178(1) (2002), 253-262.

[8] G.Horváth, J. Lawrence, L Mérai and C.Szabó, *The complexity of the equivalence problem for nonsolvable groups,* Bull. London Math. Soc. 39 (2007), 433-438.

[9] G.Horváth, C.Szabó, *The complexity of checking identities over finite groups,* Internat. J. Algebra Comput., 16(5) (2006), 931-939.

[10] H. Hunt, R. Stearns, *The complexity for equivalence for commutative rings*, Journal of Symbolic Computation, 10 (1990), 411-436.

[11] P.M.Idziak private comunication.

[12] O. Klíma, *Complexity issues of checking identities in finite monoids,* Semigroup Forum, 79(3) (2009), 435-444.

[13] B. Larose, L. Zádori, *Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras,* Internat. J. Algebra Comput., 16(3) (2006), 563-581.

[14] R.McKenzie, G.McNulty, W.Taylor, Algebras, *Lattices, and Varieties. Vol. I.* Mathematics Series, Wadsworth and Brooks/Cole, 1987.

[15] E.Post, *The Two-valued Iterative Systems of Mathematical Logic,* Annals of Mathematics Studies, N.5, Princeton University Press, 1941.

[16] T.J.Schaefer, *The complexity of satisability problems,* Proceedings of the 13th ACM Symposium on Theory of Computing, (1978), 216-226.

[17] B. Schwarz, *The Complexity of Satisfiability Problems over Finite Lattices* Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science, STACS 2004, LNCS 2996 (2004), 31-43.

Theoretical Computer Science Department,
Jagiellonian University, Krakow, Poland

tomasz.gorazd@uj.edu.pl

Institute of Computer Science,
Maria Curie-Skłodowska University, Lublin, Poland

krzacz@liza.umcs.lublin.pl