Theoretical Computer Science 407 (2008) 591–595

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Note

An assertion concerning functionally complete algebras and NP-completeness

Gábor Horváth [a], Chrystopher L. Nehaniv [a,*], Csaba Szabó [b]

[a] Centre for Computer Science & Informatics Research, University of Hertfordshire, College Lane, Hatfield, Hertfordshire AL10 9AB, United Kingdom

[b] Department of Algebra and Number Theory, Eötvös Loránd University, Budapest, Pázmány Péter sétány 1/c, 1117, Hungary

ARTICLE INFO

Article history:
Received 7 June 2007
Received in revised form 29 February 2008
Accepted 25 August 2008
Communicated by M. Ito

Keywords:
Functionally complete algebras
Identity checking
Solvability of equations
Solvability of systems of equations
NP-completeness
coNP-completeness

ABSTRACT

In a paper published in *J. ACM* in 1990, Tobias Nipkow asserted that the problem of deciding whether or not an equation over a nontrivial functionally complete algebra has a solution is NP-complete. However, close examination of the reduction used shows that only a weaker theorem follows from his proof, namely that deciding whether or not a *system* of equations has a solution is NP-complete over such an algebra. Nevertheless, the statement of Nipkow is true as shown here. As a corollary of the proof we obtain that it is coNP-complete to decide whether or not an equation is an identity over a nontrivial functionally complete algebra.

1. Introduction

The two-element Boolean algebra, finite simple non-Abelian groups or finite matrix rings over finite fields are fundamental in the theory of computation. A common property of these algebras is functional completeness. By a *functionally complete* algebra **A** we mean a finite algebra with underlying set A and with basic operations $f_1, \ldots, f_k$ such that for every nonnegative integer n and for every function $f : A^n \to A$ there is a polynomial expression $p(x_1, \ldots, x_n)$ over **A** such that for every n-tuple $(a_1, \ldots, a_n) \in A^n$ we have $p(a_1, \ldots, a_n) = f(a_1, \ldots, a_n)$. Characterizing computational complexity of different problems concerning equations, system of equations, or identities over functionally complete algebras is therefore of natural importance.

One of the oldest algebraic questions, equally important in computer science, is to decide whether or not an equation has a solution over an algebra **A**. This is called the *equation solvability problem*. For every finite algebra this problem is in NP: one substitution is enough to prove that the equation has a solution. The *system of equation solvability problem* asks whether a system of equations has a solution.

The *polynomial equivalence problem* asks if two polynomials attain the same value for *every* substitution. If neither polynomial contains constants from the algebra, then we call it the *term equivalence problem* or shortly the *equivalence problem*. The equivalence and the polynomial equivalence problems are always in coNP: one substitution is enough to prove that the two sides do not always coincide. If the two sides coincide for every substitution over **A**, then we say that the equation is an identity over **A**. Sometimes these two types of equivalence problems are called *identity checking* problems.

These complexity questions have been investigated for several types of finite algebras. Some of the most well-known results are that for the two-element Boolean algebra the equation solvability (or SAT) and system of equations solvability

* Corresponding author. Tel.: +44 1707 284470; fax: +44 1707 284303.
E-mail addresses: G.Horvath@herts.ac.uk (G. Horváth), C.L.Nehaniv@herts.ac.uk (C.L. Nehaniv), csaba@cs.elte.hu (Cs. Szabó).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.08.028

problems are NP-complete. Moreover, the equivalence and polynomial equivalence problems are coNP-complete (see e.g. [3]).

Early investigations into the equivalence problem for various finite algebraic structures were carried out by computer scientists, in particular at Syracuse University where the terminology *the term equivalence problem* was introduced. Mainly they considered finite commutative rings and finite lattices. In the early 1990s it was shown by Hunt and Stearns (see [7]) that the equivalence problem for a finite commutative ring either has polynomial time complexity or is coNP-complete. Later Burris and Lawrence proved in [1] that the same holds for finite rings in general.

The equivalence problem for finite groups has proved to be far more challenging than for finite rings. In 2004 Burris and Lawrence [2] proved that the equivalence problem for a group $\mathbf{G}$ is in P if $\mathbf{G}$ is nilpotent or $\mathbf{G} \simeq D_n$, the dihedral group, for odd n. Horváth, Lawrence, Mérai and Szabó proved that the equivalence problem is coNP-complete for non-solvable groups [5]. Goldmann and Russell [4] showed that the equation solvability problem for a group $\mathbf{G}$ is in P if $\mathbf{G}$ is nilpotent and NP-complete if $\mathbf{G}$ is non-solvable. Later Horváth and Szabó [6] presented another method, and they proved that the equivalence problem is in P for certain meta-Abelian groups. For the system of equations solvability problem a clear dichotomy holds [4, 10], with the problem being in P for Abelian groups and being NP-complete for non-Abelian groups.

Interest in the computational complexity of the equivalence problem and of the equation solvability problem for finite algebraic structures has been steadily increasing in recent years. There are many complexity results for these problems for finite monoids and semigroups [9,13,14], where the initial approach came from the complexity of recognizing formal languages. The first hardness result for semigroups was proved by Popov and Volkov [15], and several results were proved by Seif and Szabó in [12]. The case of commutative semigroups was characterized by Kisielewicz in [8].

In this paper we consider the above-mentioned complexity questions for functionally complete algebras. In Theorem 6 on page 752 of [11] Tobias Nipkow asserted the following:

Theorem 1. *The problem of deciding whether an equation over a nontrivial functionally complete algebra* $\mathbf{A}$ *has a solution is NP-complete.*

In the proof he claims to give a polynomial reduction from deciding whether or not an equation over $\mathbf{Z}_2 = (\{0, 1\}, +, \cdot)$ has a solution (a problem which is well known to be NP-complete, see e.g. [3]) to the problem of whether an equation over $\mathbf{A}$ has a solution. Following the original proof from [11] shows that Nipkow's construction actually yields a reduction to the problem of whether a *system* of equations over $\mathbf{A}$ has a solution, which proves a weaker Theorem:

Theorem 2. *The problem of deciding whether a system of equations over a nontrivial functionally complete algebra* $\mathbf{A}$ *has a solution is NP-complete.*

Theorem 2 follows from a paper of Larose and Zádori [10], too.

In Section 3 we first show why the original proof from [11] yields to Theorem 2 rather than Theorem 1. Then in Section 4 we prove Theorem 1 and the following corollary of our proof:

Theorem 3. *The polynomial equivalence problem over a nontrivial functionally complete algebra* $\mathbf{A}$ *is coNP-complete.*

2. Preliminaries

Let $\mathbf{A}$ be an algebra with underlying set A. Let p and q be two n-variable *polynomial expressions* over $\mathbf{A}$, i.e. expressions built up from variables, constants from $\mathbf{A}$ and the basic operations of $\mathbf{A}$ using composition. Whenever a polynomial expression does not contain constants from $\mathbf{A}$, we call it a *term expression*. We call the equation $p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$ an *identity* over the algebra $\mathbf{A}$ if for every n-tuple $(a_1, \ldots, a_n) \in A^n$ the equation $p(a_1, \ldots, a_n) = q(a_1, \ldots, a_n)$ holds over $\mathbf{A}$.

Our leading reference on computational complexity will be [3]. Every algebra is given by its underlying set and with the operation table of its basic operations. An instance of the *equation solvability problem* over the algebra $\mathbf{A}$ consists of two polynomial expressions p and q over $\mathbf{A}$. The question is whether or not there exists a substitution of the variables over $\mathbf{A}$ such that the two polynomials attain the same value (i.e. the equation $p = q$ has a solution over $\mathbf{A}$). Similarly, the *system of equations solvability problem* has as instance a natural number n and polynomials $p_1, \ldots, p_n, q_1, \ldots, q_n$ over $\mathbf{A}$. The question is whether there exists a substitution of variables over $\mathbf{A}$ such that $p_i = q_i$ for every $1 \leq i \leq n$. Finally, an instance of the *polynomial equivalence problem* over an algebra $\mathbf{A}$ consists of two polynomial expressions p and q over $\mathbf{A}$. The question is whether the two polynomials attain the same value for *every* substitution of the variables over $\mathbf{A}$ (i.e. the equation $p = q$ is an identity over $\mathbf{A}$). Whenever the two polynomial expressions are term expressions (i.e. expressions built up from variables using the basic operations of $\mathbf{A}$) then we call it the *term equivalence problem* or shortly the *equivalence problem*.

We define the length of a *polynomial expression* over an algebra $\mathbf{A} = (A, f_1, \ldots, f_k)$ (i.e. an expression which can be composed from the basic operations and some constants from A) in a natural way. We give a definition which represents the idea that the length is the number of characters we need to have, where every variable and constant takes one character to write (this is the unit). Denote the length of a polynomial expression $p(x_1, \ldots, x_n)$ with $\|p(x_1, \ldots, x_n)\|$.

The *length* of a polynomial expression over $\mathbf{A}$ is defined recursively:

(1) The length of a variable x or a constant c is 1: $\|x\| = \|c\| = 1$.

(2) For an m-variable basic function f of $\mathbf{A}$ and for polynomial expressions $p_1, \ldots, p_m$, the length of $f(p_1, \ldots, p_m)$ is the sum of the lengths of p_i's with an additional constant depending only on f: $\|f(p_1, \ldots, p_m)\| = c_f + \sum_{i=1}^{m} \|p_i\|$. Then the length of $f(x_1, \ldots, x_m)$ is $\|f\| = c_f + m$. (c_f represents the length of all the brackets, commas and those characters which we identify f with.)

We give some examples illustrating the general definition.

(1) For a finite group $\mathbf{G}$ let $c_f = 4$ for the group-multiplication f and let $c_g = 3$ for the group-inverse g. Now the expression $x \cdot y^{-1} \cdot z = f(f(x, g(y)), z)$ has length $2c_f + c_g + 3 = 14$.
(2) For a finite ring $\mathbf{R}$ let $c_f = c_g = 4$ for the ring-addition f and ring-multiplication g. Now the expression $x + y \cdot z = f(x, g(y, z))$ has length $c_f + c_g + 3 = 11$.
(3) If $c_f = 0$ for every basic function, then the length of a polynomial p is exactly the number of variables and constants (including multiplicities) occurring in p.

An immediate consequence of the definition is the following lemma:

Lemma 4. *For polynomial expressions $p, q_1, \ldots, q_n$ we have that*

$$\|p(q_1, \ldots, q_n)\| \leq \|p\| \cdot \max\{\|q_i\| : i = 1, \ldots, n\}.$$

Proof. Let q be a polynomial expression for which $\|q\|$ is maximal among the q_i's. Then

$$\|p(q_1, \ldots, q_n)\| = c_f + \sum_{i=1}^{n} \|q_i\| \leq c_f + n \cdot \|q\| \leq (c_f + n) \cdot \|q\| = \|p\| \cdot \|q\|. \quad \square$$

3. Proof of Theorem 2

Let $\mathbf{A}$ be a nontrivial functionally complete algebra ($|\mathbf{A}| \geq 2$). The problem is in NP, since we only need to substitute a possible solution.

It is well known (see, e.g. [3] p. 251, problem AN9) that deciding whether an equation over $\mathbf{Z}_2 = (\{0, 1\}, +, \cdot)$ has a solution is NP-complete (it is almost the same as the SAT problem). Following the proof in [11] we give a polynomial reduction from the problem of determining whether an equation over $\mathbf{Z}_2$ has a solution to the problem of deciding whether a system of equations over $\mathbf{A}$ has a solution.

Let $f(\underline{x}) = g(\underline{x})$ be an equation over $\mathbf{Z}_2$, where f and g are polynomial expressions and $\underline{x}$ is an n-tuple of free variables. We create a system of equations over $\mathbf{A}$ in polynomial time such that the system has a solution over $\mathbf{A}$ if and only if $f = g$ has a solution over $\mathbf{Z}_2$. The size of the system will be polynomial in $\|f\| + \|g\|$.

Let us denote two arbitrary distinct elements of $\mathbf{A}$ with $0_\mathbf{A}$ and $1_\mathbf{A}$. Since $\mathbf{A}$ is functionally complete, there exist two 2-variable polynomial expressions (let us denote them with $+_\mathbf{A}$ and $\cdot_\mathbf{A}$) such that $0_\mathbf{A}$ and $1_\mathbf{A}$ behave under the operations $+_\mathbf{A}$ and $\cdot_\mathbf{A}$ as 0 and 1 behave under the operations $+$ and $\cdot$, namely:
$+_\mathbf{A}(0_\mathbf{A}, 0_\mathbf{A}) = +_\mathbf{A}(1_\mathbf{A}, 1_\mathbf{A}) = 0_\mathbf{A}$, $+_\mathbf{A}(0_\mathbf{A}, 1_\mathbf{A}) = +_\mathbf{A}(1_\mathbf{A}, 0_\mathbf{A}) = 1_\mathbf{A}$,
$\cdot_\mathbf{A}(0_\mathbf{A}, 0_\mathbf{A}) = \cdot_\mathbf{A}(0_\mathbf{A}, 1_\mathbf{A}) = \cdot_\mathbf{A}(1_\mathbf{A}, 0_\mathbf{A}) = 0_\mathbf{A}$, and $\cdot_\mathbf{A}(1_\mathbf{A}, 1_\mathbf{A}) = 1_\mathbf{A}$.

There might exist several different polynomials for $+_\mathbf{A}$. Similarly there might exist several different polynomials for $\cdot_\mathbf{A}$. We choose $+_\mathbf{A}$ and $\cdot_\mathbf{A}$ arbitrarily (with respect to these properties) and fix them for the proof.

There exists a 1-variable polynomial expression $\chi_{1_\mathbf{A}}$ such that $\chi_{1_\mathbf{A}}(1_\mathbf{A}) = 1_\mathbf{A}$ and $\chi_{1_\mathbf{A}}(a) = 0_\mathbf{A}$ for every $a \neq 1_\mathbf{A}$. Now using $+_\mathbf{A}$ and $\cdot_\mathbf{A}$ instead of $+$ and $\cdot$ and using $\chi_{1_\mathbf{A}}(x_i)$ instead of the variable x_i we can encode the equation $f = g$ over $\mathbf{Z}_2$ as an equation $f_\mathbf{A} = g_\mathbf{A}$ over $\mathbf{A}$ such that $f = g$ has a solution over $\mathbf{Z}_2$ if and only if $f_\mathbf{A} = g_\mathbf{A}$ has a solution over $\mathbf{A}$. We can observe though that if we want to express this equation using the basic operations of $\mathbf{A}$ then the length of the resulting equation might be exponential in the size of the original equation (e.g. if any variable occurs more than once in the polynomial expression for $+_\mathbf{A}$ or for $\cdot_\mathbf{A}$).[1] For this reason, the proof is not a polynomial reduction from deciding whether an equation over $\mathbf{Z}_2$ has a solution to deciding whether an equation over $\mathbf{A}$ has a solution. However, using an easy trick we can encode the original equation into a *system of equations* with polynomial size in $\|f\| + \|g\|$:

At first we have the equation $f(\underline{x}) = g(\underline{x})$ over $\mathbf{Z}_2$. In every step we will shorten this equation and add other equations to our system until the equation cannot be shortened any more. In each step we search reading from left to right in our modified equation for any occurrence of $x + y$ or of $x \cdot y$, where x and y are variables or constants (polynomial expressions with length 1). If we find an occurrence of $x + y$ with variables or constants x, y then for a new variable z we replace every occurrence of $x + y$ with z in the modified equation and add the equation $z = +_\mathbf{A}(x, y)$ to our system of equations. Similarly, if we find an occurrence of $x \cdot y$ with variables or constants x, y then for a new variable z we replace every occurrence of $x \cdot y$ with z in the modified equation and add the equation $z = \cdot_\mathbf{A}(x, y)$ to our system of equations. Each step takes at most $\|f\| + \|g\|$ time and each step shortens the equation $f = g$, hence the algorithm stops in at most $(\|f\| + \|g\|)^2$ time. After

[1] An easy example for such an exponential blowup is if for a group one wants to express the commutator expression $[[[[x_1, x_2], x_3] \ldots], x_n]$ using only the inverse operation and the multiplication of the group.

the final step, in every equation of the system for every *original* variable x_i (i.e. which occurred in $f = g$) we replace x_i with $\chi_{1_{\mathbf{A}}}(x_i)$.

After this translation we have a system of equations over $\mathbf{A}$ such that the system has a solution over $\mathbf{A}$ if and only if the original equation $f = g$ had a solution over $\mathbf{Z}_2$. The size of the system is linear in the size of the equation $f = g$ over $\mathbf{Z}_2$, since there are at most $(\|f\| + \|g\|)$-many equations, and by Lemma 4 each equation has length at most $(\|+_{\mathbf{A}}\| + \|\cdot_{\mathbf{A}}\|) \cdot \left\|\chi_{1_{\mathbf{A}}}\right\|$, which does not depend on the equation but only on the algebra $\mathbf{A}$. The time of the translation of $f = g$ over $\mathbf{Z}_2$ to a system of equations over $\mathbf{A}$ is polynomial as well, which finishes the proof.

4. Proof of Theorem 1 and a consequence

Let $\mathbf{A}$ be a nontrivial functionally complete algebra ($|\mathbf{A}| \geq 2$). The problem is in NP, since we only need to substitute a possible solution.

It is well known (see, e.g. [3]) that deciding whether a formula written in conjunctive normal form can be satisfied over the two-element Boolean algebra $\mathbf{B} = (\{0, 1\}, \neg, \vee, \wedge)$ is NP-complete (this is called the SAT problem). The formula is usually given by the clauses, which we take the conjunctions of, where each clause is a disjunction of arbitrary many literals, i.e. variables or negations of variables ([3] p. 259 problem LO1). The problem remains NP-complete, if every clause in the conjunctive normal form contains exactly 3 literals (this is called the 3SAT problem, [3] p. 259 problem LO2). We will give a polynomial reduction from the problem of determining whether a 3SAT formula can be satisfied over $\mathbf{B}$ to the problem of whether an equation over $\mathbf{A}$ has a solution.

Let $\varphi(\underline{x}) = \bigwedge_{i=1}^{n} p_i$ be a 3SAT formula over $\mathbf{B}$. We create an equation over $\mathbf{A}$ such that the equation has a solution over $\mathbf{A}$ if and only if φ can be satisfied over $\mathbf{B}$. The length of the equation will be polynomial in the size of the formula.

Let us denote two arbitrary distinct elements of $\mathbf{A}$ with $0_{\mathbf{A}}$ and $1_{\mathbf{A}}$. Since $\mathbf{A}$ is functionally complete, there exists a 2-variable polynomial expression $\wedge_{\mathbf{A}}$ such that $0_{\mathbf{A}}$ and $1_{\mathbf{A}}$ behave under the operation $\wedge_{\mathbf{A}}$ as 0 and 1 behave under the operation $\wedge$, namely $\wedge_{\mathbf{A}}(0_{\mathbf{A}}, 0_{\mathbf{A}}) = \wedge_{\mathbf{A}}(0_{\mathbf{A}}, 1_{\mathbf{A}}) = \wedge_{\mathbf{A}}(1_{\mathbf{A}}, 0_{\mathbf{A}}) = 0_{\mathbf{A}}$, and $\wedge_{\mathbf{A}}(1_{\mathbf{A}}, 1_{\mathbf{A}}) = 1_{\mathbf{A}}$. There might exist several different polynomials for $\wedge_{\mathbf{A}}$. We choose $\wedge_{\mathbf{A}}$ arbitrarily (with respect to these properties) and fix it for the proof. Similarly, for each of the eight possible 3-variable forms of disjunctive clause $q_j = q_j(x_1, x_2, x_3)$, $(j = 1, \ldots, 8)$ we can choose an arbitrary but fixed 3-variable polynomial expression $q_{j,\mathbf{A}}$ such that $0_{\mathbf{A}}$ and $1_{\mathbf{A}}$ behave under the function $q_{j,\mathbf{A}}$ as 0 and 1 behave under the clause q_j. Moreover there exists a 1-variable polynomial expression $\chi_{1_{\mathbf{A}}}$ such that $\chi_{1_{\mathbf{A}}}(1_{\mathbf{A}}) = 1_{\mathbf{A}}$ and $\chi_{1_{\mathbf{A}}}(a) = 0_{\mathbf{A}}$ for every $a \neq 1_{\mathbf{A}}$.

For every positive integer number k we will use a polynomial $\wedge^{(k)} = \wedge_{\mathbf{A}}^{(k)}(x_1, \ldots, x_k)$ over $\mathbf{A}$ in a way that it behaves on inputs from $\{0_{\mathbf{A}}, 1_{\mathbf{A}}\}$ the very same as $\bigwedge_{i=1}^{k} x_i$ behaves on the inputs $\{0, 1\}$ over $\mathbf{B}$. Let us define $\wedge^{(k)}$ in the following way: Let $\wedge_{\mathbf{A}}^{(1)}(x_1) = x_1$ and $\wedge_{\mathbf{A}}^{(2)}(x_1, x_2) = \wedge_{\mathbf{A}}(x_1, x_2)$. For every integer $i \geq 2$ let

$$\wedge_{\mathbf{A}}^{(2i-1)}(x_1, \ldots, x_{2i-1}) = \wedge_{\mathbf{A}}^{(2)}\left(\wedge_{\mathbf{A}}^{(i)}(x_1, \ldots, x_i), \wedge_{\mathbf{A}}^{(i-1)}(x_{i+1}, \ldots, x_{2i-1})\right),$$

$$\wedge_{\mathbf{A}}^{(2i)}(x_1, \ldots, x_{2i}) = \wedge_{\mathbf{A}}^{(2)}\left(\wedge_{\mathbf{A}}^{(i)}(x_1, \ldots, x_i), \wedge_{\mathbf{A}}^{(i)}(x_{i+1}, \ldots, x_{2i})\right).$$

It is clear that $\wedge_{\mathbf{A}}^{(k)}$, for every integer k, has the required property.

Now using the expression $q_{j,\mathbf{A}}$ instead of the clause q_j, using $\wedge_{\mathbf{A}}^{(n)}$ instead of $\wedge_{i=1}^{n}$ and using $\chi_{1_{\mathbf{A}}}(x_i)$ instead of the variable x_i we can encode the formula φ over $\mathbf{B}$ as an expression $\varphi_{\mathbf{A}}$ over $\mathbf{A}$ such that φ can be satisfied over $\mathbf{B}$ if and only if $\varphi_{\mathbf{A}} = 1_{\mathbf{A}}$ has a solution over $\mathbf{A}$. The only remaining part is to prove that $\|\varphi_{\mathbf{A}}\|$ is polynomial in $\|\varphi\|$.

Let $c = \left\|\chi_{1_{\mathbf{A}}}\right\|$, let $l = \|\wedge_{\mathbf{A}}\|$ and let $d = \max\left\{\left\|q_{j,\mathbf{A}}\right\| : j = 1, \ldots, 8\right\}$ the length of the longest clause expression. For every k we have $\left\|\wedge_{\mathbf{A}}^{(k)}\right\| \leq l^{\lceil \log k \rceil} \leq l \cdot k^{\log l}$, which is quite straightforward from $\left\|\wedge_{\mathbf{A}}^{(k)}\right\| \leq \left\|\wedge_{\mathbf{A}}^{(2)}\right\| \cdot \max\left\{\left\|\wedge_{\mathbf{A}}^{(\lceil k/2 \rceil)}\right\|, \left\|\wedge_{\mathbf{A}}^{(\lfloor k/2 \rfloor)}\right\|\right\}$. By log we mean the base 2 logarithm function.

Using Lemma 4 we can conclude that the length of the expressed 3SAT formula $\varphi_{\mathbf{A}}$ over $\mathbf{A}$ is not more than $c \cdot d \cdot l \cdot n^{\log l}$, which is polynomial in the length of the original 3SAT formula $\|\varphi\|$, since $n \leq \|\varphi\|$ and c, d, l depend only on $\mathbf{A}$. Thus, Theorem 1 is proved.

With a slight modification we can easily prove Theorem 3. Let $\mathbf{A}$ be a nontrivial functionally complete algebra ($|\mathbf{A}| \geq 2$). The problem is in coNP, since we only need to substitute a possible counterexample.

In the proof of Theorem 1, for every 3SAT formula φ we created a polynomial expression $\varphi_{\mathbf{A}}$ over $\mathbf{A}$ such that φ can be satisfied over $\mathbf{B}$ if and only if $\varphi_{\mathbf{A}} = 1_{\mathbf{A}}$ has a solution over $\mathbf{A}$. Moreover the length of $\varphi_{\mathbf{A}}$ was polynomial in the length of φ. Let us observe that the image of $\varphi_{\mathbf{A}}$ over $\mathbf{A}$ is a (not necessarily proper) subset of $\{0_{\mathbf{A}}, 1_{\mathbf{A}}\}$, hence $\varphi_{\mathbf{A}} = 1_{\mathbf{A}}$ has a solution over $\mathbf{A}$ if and only if $\varphi_{\mathbf{A}} = 0_{\mathbf{A}}$ is *not* an identity over $\mathbf{A}$. This is a polynomial reduction from the problem of 3SAT over $\mathbf{B}$ to the problem of determining whether an equation is an identity over $\mathbf{A}$.

Acknowledgements

The research of the third author and, in part, of the first author was supported by the Hungarian National Foundation for Scientific Research, Grants N67867 and K67870.

References

[1] S. Burris, J. Lawrence, The equivalence problem for finite rings, Journal of Symbolic Computation 15 (1993) 67–71.
[2] S. Burris, J. Lawrence, Results on the equivalence problem for finite groups, Algebra Universalis 52 (4) (2004) 495–500. 2005.
[3] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman & Co, San Francisco, 1979.
[4] M. Goldmann, A. Russell, The complexity of solving equations over finite groups, in: Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, May, 1999, pp. 80–86.
[5] G. Horváth, J. Lawrence, L. Mérai, Cs. Szabó, The complexity of the equivalence problem for non-solvable groups, Bulletin of the London Mathematical Society 39 (3) (2007) 433–438.
[6] G. Horváth, Cs. Szabó, The complexity of checking identities over finite groups, International Journal of Algebra and Computation 16 (5) (2005) 931–940.
[7] H. Hunt, R. Stearns, The complexity for equivalence for commutative rings, Journal of Symbolic Computation 10 (1990) 411–436.
[8] A. Kisielewicz, Complexity of semigroup identity checking, International Journal of Algebra and Computation 14 (4) (2004) 455–464.
[9] O. Klíma, Unification modulo associativity and idempotency', Ph.D. Thesis, Masarik University, Brno, 2004.
[10] B. Larose, L. Zádori, Taylor terms, constraint satisfaction and the complexity of polynomial equations over finite algebras, International Journal of Algebra and Computation 16 (3) (2006) 563–581.
[11] T. Nipkow, Unification in primal algebras, their powers and their varieties, Journal of the Association for Computing Machinery 37 (1) (1990) 742–776.
[12] S. Seif, Cs. Szabó, Computational complexity of checking identities in 0-simple semigroups and matrix semigroups over finite fields, Semigroup Forum 72 (2) (2006) 207–222.
[13] P. Tesson, Computational complexity questions related to finite monoids and semigroups, Ph.D. Thesis, McGill University, Montreal, 2004.
[14] P. Tesson, D. Therien, Monoids and Computations, International Journal of Algebra and Computation 14 (5–6) (2004) 801–816.
[15] M.V. Volkov, Checking identities in semigroups, Lecture presented at the Conference on Universal Algebra, Nashville, 2002.