Annales UMCS Informatica AI 5 (2006) 227-236

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Counting solutions of equations over two-element algebras

Jacek Krzaczkowski*

Institute of Computer Science, Maria Curie-Skłodowska University,
Pl. M. Curie-Skłodowskiej 5, 20-031 Lublin, Poland

Abstract

Solving equations is one of the most important problems in computer science. Apart from the problem of existence of solutions of equations we may consider the problem of a number of solutions of equations. Such a problem is much more difficult than the decision one. This paper presents a complete classification of the complexity of the problem of counting solutions of equations over any fixed two-element algebra. It is shown that the complexity of such problems depends only on the clone of term operations of the algebra and for any fixed two-element algebra such a problem is either in FP or #Pcomplete.

1. Introduction

One of the most important questions in computer science is if P is equal to NP. It is well known that if $P \neq NP$ then in NP there exists a problem which is neither in P nor NP-complete. Independently of the fact whether P is equal to NP or not, there exist numerous natural classes of problems such that every problem in such a class is either P or NPC. One of the best known such classes of problems is the Constraint Satisfaction Problem over a two-element [1] and three-element [2] domains. Another such a class of problems is solving equations over two-element algebras [3].

Similarly, there exist classes of counting problems whose members are either in FP or #P-complete (e.g. #CSP over two- and three-element domain [4,5]). This paper presents the classification of complexity of problems of counting solutions of equations over fixed two-element algebra which is one of such classes of problems.

Another interesting fact is that if we fix a two-element algebra **A**, then the complexity of problems considered in this paper can be deduced from a termal clone of **A**. The same feature is exhibited by the decision problem of existence of solution of equations over fixed two-element algebra [3]. Unfortunately, the

*E-mail address: krzacz@ii.uj.edu.pl

computational complexity of solving equations over a fixed finite algebra in general does not depend only on termal clone of this algebra.

Example 1.1. *Solving equations of polynomials over* $(S_3, \circ)$ *is in P ([6]) and the same problem over* $(S_3, \circ, [,])$*, where* $[x, y] = x^{-1} \circ y^{-1} \circ x \circ y$*, is NP-complete ([7]). Of course,* $Clo(S_3, \circ) = Clo(S_3, \circ, [,])$.

2. Counting class #P and #P complete problems

#P as the class of the computational complexity of counting problems was introduced by Valiant in [8]. We assume the following definition of #P (this definition comes from [9]):

Definition 2.1. *Counting problem connected with a binary relation Q is a problem of a number of y, such that* $(x,y) \in Q$ *for given x.*

#P is a class of counting problems connected with any binary relation Q fulfilling the following conditions:

- *there is a polynomial-time algorithm to determine, for given x and y, if* $(x, y) \in Q$,
- *there exists a constant* $k \in \mathbb{N}$*, such that for all* $(x, y) \in Q$, $|y| \leq |x|^k$.

To define #P-completeness we need a definition of reduction. We assume the definition from [9]:

Definition 2.2. *Let A and B belong to #P. The reduction from A to B is called a pair of functions R and S such that:*

- *for every x an instance of the problem A,* $R(x)$ *is an instance of the problem B,*
- *if N is a proper answer for the problem B with the instance* $R(x)$*, then S(N) is a proper answer for the problem A with the instance x.*

In the end of this section we define what it means that the problem is #P-complete.

Definition 2.3. *The problem* $A \in \#P$ *is #P-complete if for all problems* $B \in \#P$ *there is a reduction from B to A.*

3. Definitions

In this paper we use the following standard definition of terms over the algebra $\mathbf{A} = (A, f_1, \ldots, f_k)$:

- variables are terms,
- if $T_1, \ldots, T_n$ are terms and $i \in \{1, \ldots, k\}$, then $f_i(T_1, \ldots, T_n)$ is a term.

Similarly, we define polynomials:

- variables are polynomials,

- constants are polynomials,
- if $P_1,\ldots, P_n$ are polynomials and $i\in\{1,\ldots, k\}$, then $f_i(P_1,\ldots, P_n)$ is a polynomial.

A clone on a set **A** is a set of finitary operations on **A** that contains the projection and is closed under composition of operations.

Clo(**A**) denoted the clone of term operations on the algebra **A**, i.e. the clone generated by the set of basic operations of **A**.

Pol(**A**) denoted the clone of polynomial operations of the algebra **A**, i.e. the clone generated by the set of basic operations of **A** along with all of the constant operations on the set **A**.

Definition 3.1. *A problem for an algebra **A** will be called*

- ***clone-determined** iff for every algebra **B** such that Clo(**A**) = Clo(**B**) the problem for **A** and **B** is polynomially equivalent.*
- ***not clone-determined** otherwise.*

Clones on the fixed set constitute a lattice. The clones on the two-element set were classified by Post, see [10]. We use the original Post's notation for clones.

Obviously, every term T is equivalent to some operation t from *Clo*(**A**) and every polynomial P is equivalent to some operation p from *Pol*(**A**). We say that equation between terms over $\mathbf{A} = (A, f_1,\ldots, f_k)$ in the form $T_1(x_1,\ldots, x_n) = T_2(x_1,\ldots, x_n)$ is satisfiable iff there a exists function s: $\{x_1,\ldots, x_n\}\to$A, such that $t_1(s(x_1),\ldots, s(x_n)) = t_2(s(x_1),\ldots, s(x_n))$, where t_1, $t_2\in Clo(\mathbf{A})$ are operations equivalent to terms T_1 and T_2. Such a function s is called solution. We may similarly define the solution for equations between polynomials and equations between a term and a constant.

In this paper we will consider problems where the question is how many solutions the given equation has got. Defining three main problems we consider.

Definition 3.2. #TERM-SAT(A) *is a problem of a number of solutions of equations between term over an algebra A.*

Definition 3.3. #POL-SAT(***A***) *is a problem of a number of solutions of equations between polynomials over an algebra **A**.*

Definition 3.4. #$\mathrm{TERM_C}$-SAT(***A***) *is a problem of a number of solutions of equations between term and constant over an algebra **A**.*

We will prove that #TERM-SAT(A), #POL-SAT(A), #$\mathrm{TERM_C}$-SAT(A) for a two-element algebra A is clone-determined. Moreover, it is either #P-complete or FP.

4. #TERM-SAT

In this section we will show a main theory of our paper which gives us a classification of a complexity of #TERM-SAT(A) for any two-element algebra.

All proofs of #P-completeness of the considered problems for some classes of algebras in this paper use the fact that there exist terms over these algebras equivalent to some special functions (for example $\vee$, $\wedge$ or $\neg$). But the fact that there exists the term equivalent to such functions is not sufficient – this term must have a suitable form. The following example is a good illustration of this fact:

Example 4.1. *Consider an algebra (2,NAND) and the following terms:*

- *$t(x,y) = (x \text{ NAND } y) \text{ NAND } (x \text{ NAND } y) \equiv x \wedge y$,*
- *$h(x,y,z) = (x \text{ NAND } y) \text{ NAND } (z \text{ NAND } (z \text{ NAND } z)) \equiv x \wedge y$ (h does not depend on z).*

Now, if we want to express the function $x_1 \wedge x_2 \wedge \ldots \wedge x_n$ using the term t we obtain an expotentially longer expression, but if we use the term h the obtained expression will be only linear longer than the original one.

In the following there is a very useful definition:

Definition 4.2. *Let $T(x_1,\ldots, x_n)$ be a term over an algebra **A** and t be a function belonging to Clo(**A**).*
We say that T is frugal iff for all x_i, such that $t(x_1,\ldots, x_n)$ depends on x_i, x_i occurs in T only once.

Moreover, we say that the function $f \in Clo(\mathbf{A})$ is frugally definable over **A** if there exist any frugal term T over **A** which is equivalent to f. Such a term will be called a frugal definition of f. Observe that if in a term containing functions $f_1,\ldots, f_n$ any of these functions will be replaced by an equivalent frugal term, the obtained term will be equivalent to the original one and at most polynomially longer.

To prove the main theory of this paper we will need the following two lemmas:

Lemma 4.3. *If $Clo(2, f_1,\ldots, f_n) = Clo(2,\wedge,\neg)$, then $\wedge$, $\vee$ and $\neg$ are frugally definable over $(2, f_1,\ldots, f_n)$.*

Lemma 4.4. *If $Clo(2, f_1,\ldots, f_n) = Clo(2,\wedge,\vee,0,1)$, then $\wedge$ and $\vee$ are frugally definable over $(2, f_1,\ldots, f_n)$.*

Proofs of these two lemmas are a simple conclusion from the proofs of Theorem 7 and Theorem 16 in [3].

Lemma 4.5. Let A be a two-element algebra such that one of the following holds:

- $F_5^\infty = Clo(2, ki) \subset Clo(\mathbf{A})$,
- $F_1^\infty = Clo(2, ki_{dual}) \subset Clo(\mathbf{A})$

then #TERM-SAT(***A***) *is #P-complete.*
$ki(x, y, z) = x \wedge (y \to z)$
$ki_{dual}(x, y, z) = \neg ki(\neg x, \neg y, \neg z)$

Proof: We will prove the first case only. The proof of the dual case is very similar.

Assume that **A** = (2, f_1,…, f_n). Observe that $Clo(2, ki, 0, 1) = Clo(2,\wedge,\neg)$ which implies that $Clo(2, f_1,\ldots, f_n, 0, 1) = Clo(2,\wedge,\neg)$. By Lemma 4.3 we have that $\wedge$, $\vee$ and $\neg$ are frugally definable over $(2, f_1,\ldots, f_n, 0, 1)$. Moreover, there exists a term over A which is equivalent to the operator $ki(x, y, z)$ but we do not now if this term is frugal or not.

Now we show the reduction from #SAT to #TERM-SAT(**A**). First, the reduction for the given CNF-formula S creates the term R by replacing in S every occurrence of $\wedge$, $\vee$, $\neg$ by frugal terms over $(2, f_1,\ldots f_n, 1, 0)$ equivalent to them. Next, the reduction replaces every occurrence of 1 and 0 in *R* by *x* and *y* (without loss of generality we may assume that *x* and *y* do not occur in *R*). Denote such a new term by *S*'. At the end the reductions return the following equation:

$$ki\left(x, \neg S'\left(x, y, z_1, \ldots, z_n\right), y\right) = x \tag{1}$$

where, in fact, instead of $\neg$ there is used a suitable term over $Clo(2, f_1,\ldots, f_n,0,1)$ with x and y in a place of 1 and 0.

The second part of the reduction, transforming N, number of solutions of the equation above, to number of assignments satisfying the formula S, working as the following:

- if N is a power of 2, then the reduction returns N/4 (observe that it is easy to check if a given number is power of 2).
- otherwise the reduction returns $N - 2^{\lfloor \log_2 N \rfloor} - 2^{\lfloor \log_2 N \rfloor - 1}$ (it may be computed in polynomial deterministic time).

To see that the presented reduction is proper, consider following three cases of solutions of the equation:

- $x = 0$ – in this case the equation is satisfied with all assignment of other variables. There are 2^{n+1} solutions with $x = 0$,
- $x = 1$ and $y = 1$ – in this case similarly to the previous one the equation is satisfied with all assignments of $z_1,\ldots, z_n$. There are 2^n solutions with $x = 1$ and $y = 1$.

– $x = 1$ and $y = 0$ – in this case the equation is satisfied iff $\neg S'(x, y, z_1,\ldots, z_n) = 0$. Notice that because $x = 1$ and $y = 0$, $\neg S'(x, y, z_1,\ldots, z_n) = 0$ iff assignment of $z_1,\ldots, z_n$ satisfies the formula S. Denote the number of assignments satisfying the formula S by M. Of course, $M \leq 2^n$.

Adding up solutions of the equation we obtain $N = 2^{n+1} + 2^n + M$. Observe that N is power of 2 iff S is a tautology and in this case $N = 2^{n+2} = 4 \cdot 2^n = 4 \cdot M$. Otherwise, $M = N - 2^{n+1} - 2^n = N - 2^{\lfloor \log_2 N \rfloor} - 2^{\lfloor \log_2 N \rfloor - 1}$.

To complete the proof, note that both parts of the reduction work in a polynomial deterministic time. ■

Lemma 4.6. *Let* $\boldsymbol{A}$ *be a two-element algebra such that one of the following holds:*

– $F_6^\infty = Clo(2, ka) \subset Clo(\mathbf{A}) \subset Clo(2, \wedge, \vee, 1, 0) = A_1$,

– $F_2^\infty = Clo(2, ka_{dual}) \subset Clo(\mathbf{A}) \subset Clo(2, \wedge, \vee, 1, 0) = A_1$

then #TERM-SAT($\boldsymbol{A}$) *is #P-complete.*

$ka(x, y, z) = x \wedge (y \vee z)$,

$ka_{dual}(x, y, z) = \neg ka(\neg x, \neg y, \neg z)$.

Proof: We will prove the first case only. The proof of the dual case is very similar.

Assume that $\mathbf{A} = (2, f_1,\ldots, f_n)$. Note that $Clo(2, ka, 1, 0) = Clo(2, \wedge, \vee, 1, 0)$ which implies that $= (2, f_1,\ldots, f_n, 1, 0) = Clo(2, \wedge, \vee, 1, 0)$. By Lemma 4.4. we have that $\wedge$ and $\vee$ are frugally definable over $(2, f_1,\ldots, f_n, 1, 0)$. Moreover, over $\mathbf{A}$ exists a term equivalent to ka, but we do not know if this term is frugal.

First, consider the following term:

$$W(z, z', t) = (z \wedge z') \vee ((z \vee z') \wedge t).$$

Observe that W depends on t only if $z \neq z'$. Obviously W is frugally definable over $(2, f_1,\ldots, f_n, 1, 0)$.

Define $W'(x, y, z, z', t)$ obtained by replacing every occurrence of 1 and 0 in W by x and y. Furthermore, we denote by $\wedge$' and $\vee$' the terms over A obtained from frugal terms over $(2, f_1,\ldots, f_n, 1, 0)$ equivalent to $\wedge$ and $\vee$, by replacing every occurrence of 1 and 0 by x and y.

Now, we are ready to define reduction from #SAT to #TERM-SAT($\mathbf{A}$). For a given CNF-formula S, the reduction returns the following equation:

$$x \wedge \left(x \vee W'\left(x, y, z_1, z_1', W'\left(\ldots W'\left(x, y, z_k, z_k', S'\left(x, y, z_1, \ldots, z_k, z_1', \ldots, z_k' \right) \right) \right) \right) \right) = x, \quad (2)$$

where $S'(x, y, z_1,\ldots, z_k, z_1',\ldots, z_k')$ is received from S by replacing every occurrence of $\wedge$, $\vee$ and $\neg z_i$ by $\wedge$', $\vee$' and z_i' for $i \in \{1,\ldots, n\}$.

Assume that N is a number of solutions of the above equation. The reduction computes a number of assignments satisfying the formula S using the following formula:

$M = N - 2^{\lfloor \log_2 N \rfloor} - 2^{\lfloor \log_2 N \rfloor - 1} - 2^{\lfloor \log_2 N \rfloor - 3} - 2^{\lfloor \log_2 N \rfloor - 4} - 2^{\lfloor \log_2 N \rfloor - 5} - \ldots - 2^{\lfloor \log_2 N \rfloor / 2 - 2}$.

Certainly, the reduction works in a polynomial deterministic time. To prove correctness of the reduction we need to consider the following five cases:

- $x = 0$ – in this case the equation is satisfied with all assignments of other variables (there exist 2^{2k+1} such solutions),
- $x = 1$ and $y = 1$ □ in this case the equation is satisfied with all assignments of other variables too (there exist 2^{2k} such solutions),
- $x = 1, y = 0, z_j = z_j' = 1$ for some j and $z_i \neq z_i$ for $1 \leq i <$ j – there are $2^{2k-2} + 2^{2k-3} + \ldots + 2^{k-1}$ such solutions of the equation,
- $x = 1, y = 0, z_j = z_j' = 0$ for some j and $z_i \neq z_i$ for $1 \leq i <$ j – in this case the equation is not satisfied.
- $x = 1$, $y = 0$ and $z_i \neq z_i$ for all $i \in \{1,\ldots, k\}$ – in this case the equation is satisfied iff $S'(x, y, z_1,\ldots, z_k, z_1',\ldots, z_k') = 1$. Because $x = 1$ and $y = 0$ and $z_i \neq z_i'$ for all i it occurs only if assignments of $z_1,\ldots, z_k$ satisfy the formula S.

Now, we leave completion of the proof to the reader. ■

Lemma 4.7. *Let* $\boldsymbol{A}$ *be a two-element algebra such that one of the following holds:*

- $Clo(\boldsymbol{A}) = Clo(2, d)$,
- $Clo(\boldsymbol{A}) = Clo(2, d, +_3)$,
- $Clo(\boldsymbol{A}) = Clo(2, d, \neg)$.

then #TERM-SAT($\boldsymbol{A}$) *is #P-complete.*

$d(x, y, x) = (x \vee y) \wedge (x \vee z) \wedge (y \vee z)$,

$+_3(x, y, z) = x + y + z$.

Proof: Assume that $\mathbf{A}=(2, f_1,\ldots, f_n)$. Obviously, $d \in Clo(\mathbf{A})$. For all $i \in \{1,\ldots, n\}$ the following holds: $f_i(x_1,\ldots, x_l) = \neg f(\neg x_1,\ldots,\neg x_l)$. It is easy to see that all $f \in Clo(\mathbf{A})$ retain this property which implies that for all terms over A we have that:

$$T(1,0,x_1,...,x_l) = 1 \Leftrightarrow T(1,0,\neg x_1,...,\neg x_l) = 0 . \tag{3}$$

Consider the following equation:

$$d\left(p,q,T\left(p,q,z_1,...,z_k\right)\right) = p \tag{4}$$

Observe that if $p = q$ then the equation is satisfied (the equation has 2^{k+1} such solutions). If $p \neq q$ then above equation is satisfied iff the following equation is satisfied:

$$T\left(p,q,z_1,...,z_k\right) = p . \tag{5}$$

We will reduce #SAT to #TERM-SAT(A). Let S be a CNF-formula. Our reduction depends on algebra A and we have to consider the following two cases:

- $Clo(2, f_1,\ldots, f_n,0,1) = Clo(2, d,0,1) = Clo(2,\wedge,\vee,0,1)$ – in this case by the Lemma 4.4. we can frugally define function t1 such that $t_1(1,0,z,w) = z \wedge w$ (observe that $t_1(0,1,z,w) = z \vee w$). The first part of the reduction for the formula S returns equation 4, where T is the term on the left side of equation 2 from the proof of Lemma 4.6. in which every occurrence of $\wedge$', $\vee$' is replaced by $t_1(p,q,z,w)$, $t_1(q,p,z,w)$.
- $Clo(2, f_1,\ldots, f_n,0,1) = Clo(2, d,+_3,0,1) = Clo(2, d,\neg,0,1) = Clo(2,\wedge,\neg)$ – in this case by Lemma 4.3. we can frugally define functions t_1 and t_2 such that $t_1(1,0,z,w) = z \wedge w$ (obviously, $t_1(0,1,z,w) = z \vee w$) and $t_2(0,1,z) = t_2(1,0,z) = \neg z$. The first part of the reduction for the formula S returns equation 4, where T is a term on the left side of equation 1 from the proof of Lemma 4.5. Similarly to the previous case, we use $t_1(p,g,z,w)$, $t_1(q,p,z,w)$, $t_2(q,p,z)$ instead of $\wedge$, $\vee$, $\neg$.

The second part of the reduction is a small modification of one in the proofs of Lemma 4.5. and Lemma 4.6. The difference is that counting solutions of the equation 4 we have to consider the case when $p = q$ and the fact that T has the same number of solutions in both cases, when $p = 1$, $q = 0$ and $p = 0$, $q = 1$. The rest of the proof is obvious and we leave it to the reader. ■

Theorem 4.8. *Let* $\mathbf{A}$ *be a two-element algebra such that one of the following holds:*

- $ka \in Clo(\mathbf{A})$,
- $ka_{dual} \in Clo(\mathbf{A})$,
- $d \in Clo(\mathbf{A})$.

then #TERM-SAT(A) (#POL-SAT(A)) *is #P-complete. Otherwise,* #TERM-SAT(A) (#POL-SAT(A)) *is in FP.*

$ka(x, y, z) = x \wedge (y \vee z)$,
$ka_{dual}(x, y, z) = \neg ka(\neg x, \neg y, \neg z)$,
$d(x, y, z) = (x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$.

Proof: First, we will prove the theorem for #TERM-SAT($\mathbf{A}$). If $ka \in Clo(\mathbf{A})$ or $ka_{dual} \in Clo(\mathbf{A})$, then the #P-completeness of #TERM-SAT($\mathbf{A}$) is implied by Lemma 4.5. and Lemma 4.6. The case if $ka \notin Clo(\mathbf{A})$ and $d \in Clo(\mathbf{A})$ is considered in Lemma 4.7. At the end, there are only few cases left and for them the proof of containing in FP is obvious.

To prove the theorem for #POL-SAT(A) where $A = (2, f_1,\ldots, f_n)$, it is enough to consider #TERM-SAT($\mathbf{A'}$), where $\mathbf{A'} = (2, f_1,\ldots, f_k,0,1)$. ■

Conclusions

Using the Theorem 4.8. we may prove easily the theorem characterizing computational complexity of #TERM$_C$-SAT(A):

Theorem 5.1. *Let **A** be a two-element algebra such that:*
- $ki \in Clo(\mathbf{A})$,
- $ki_{dual} \in Clo(\mathbf{A})$.

then #TERM$_C$-SAT(A) *is #P-complete. Otherwise* #TERM-SAT(A) *is in FP*.
$ki(x, y, z) = x \wedge (y \to z)$,
$ki_{dual}(x, y, z) = \neg ki(\neg x, \neg y, \neg z)$.

Proof: #P-completeness of the above cases is an easy conclusion from the proofs of Lemma 4.5. and Lemma 4.6. It is enough to replace by 1 the variable x on the right side of the equations returned by reductions and slightly modify algorithm computing the number of assignments satisfying the CNF-formula.

If $Clo(2, d) = Clo(\mathbf{A}) = Clo(2, d, \neg)$, then for all f basic operations **A**, we have that $f(x_1,\ldots, x_n) = \neg f(\neg x_1,\ldots,\neg x_n)$. This fact implies that every equation on the form:

$$T(x_1,...,x_k) = c\ .$$

where $c \in 2$ and T is a term over **A**, has exactly 2^{k-1} solutions. So to count solutions it is enough to count variables occurring in the equation and #TERM$_C$-SAT(**A**) is in FP.

In all other cases #TERM$_C$-SAT(**A**) is by theorem 4.8 in FP. It is because equations between a term and a constant are a special case of equations between two polynomials. So if #POL-SAT(**A**) is in FP, then #TERM$_C$-SAT(A) is in FP. ■

Similarly to the #TERM-SAT(**A**), #POL-SAT(A), #TERM$_C$-SAT(**A**) we may define similar problems for the systems of equations #STERM-SAT(**A**), #SPOL-SAT(**A**), #STERM$_C$-SAT(**A**).

Theorem 5.2. *Let **A** be a two-element algebra such that $Clo(A) \in Clo(2,+,\neg) = L_4$ then* #TERM-SAT(***A***), #POL-SAT(***A***) *and* #TERM$_C$-SAT(***A***) *are in FP. Otherwise those problems are #P-complete*.

Proof: This theorem is a simple conclusion of the results obtained by Creignou and Herman [4]. ■

From Theorem 4.8 we have that in most difficult cases in theorem 5.2, one equation it is enough to make the problem #P-complete. Only new hard cases that it is $(2,\vee)$, $(2,\vee,0)$, $(2,\vee,1)$, $(2,\vee,0,1)$, $(2,\wedge)$, $(2,\wedge,0)$, $(2,\wedge,1)$, $(2,\wedge,0,1)$. It is easy to see that in these cases if we fix the maximum allowed number of

equations in the system of equations, the problem of counting their solutions is in FP.

References

[1] Scheafer T., *The complexity of satisfiability problems*, Proceedings 10th ACM Symposium on Theory of Computing (STOC'78), (1978) 216.
[2] Bulatov A., *A dichotomy theorem for constraints on a three-element set*, Proceedings of 43rd IEEE Symposium on Foundations of Computer Science (FOCS'02), Vancouver, Canada, (2002) 649.
[3] Gorazd T.A., Krzaczkowski J., *Solving equations over two-element algebras*, manuscript, (2005).
[4] Creignou N., Hermann M., *Complexity of generalized satisfiability counting problems*, Information and Computation, 125(1) (1996) 1.
[5] Bulatov A., Dalmau V., *Towards a dichotomy theorem for the counting constraint satisfaction problem*, Technical Report PRG-RR-03-12, Computing Laboratory, University of Oxford, Oksford, UK, (2003).
[6] Horv'ath G., Szab'o C., *Checking identities in finite groups*, manuscript, (2004).
[7] Idziak P.M., private comunication.
[8] Valiant L.G., *The complexity of computing permanent*, Theoretical Computing Science, 8 (1979) 189.
[9] Papadimitrou Ch.H., *Computational Complexity*, Adison-Wesley, (1994).
[10] Post E., *The Two-valued Iterative Systems of Mathematical Logic*, Annals of Mathematics Studies, Princenton University Press, 5 (1941).