

On the Computational Complexity of Some Classical Equivalence Relations on Boolean Functions*

B. Borchert,¹ D. Ranjan,² and F. Stephan¹

¹Mathematisches Institut, INF 294,
Universität Heidelberg, 69120 Heidelberg, Germany
{bb,fstephan}@math.uni-heidelberg.de

²Department of Computer Science, Science Hall 159,
New Mexico State University,
Las Cruces, NM 88011, USA
dranjan@cs.nmsu.edu

Abstract. The paper analyzes in terms of polynomial time many-one reductions the computational complexity of several natural equivalence relations on Boolean functions which derive from replacing variables by expressions, one of them is the Boolean isomorphism relation. Most of these computational problems turn out to be between co-NP and Σ_2^P .

1. Introduction

It is easy to see that the following computational problem is co-NP-complete: Given two Boolean circuits $f(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n)$, do they represent the same Boolean function? An extension of this problem is the following problem: Given two Boolean circuits $f(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n)$, are the two Boolean functions isomorphic, i.e., can one permute the variables of the second such that afterward they represent the same Boolean function? The two Boolean functions given by $x \wedge \neg y$ and $y \wedge \neg x$ are an example for Boolean functions which are isomorphic but not identical. We give some evidence for the hypothesis that it is more difficult to find out whether the Boolean functions given by

* The third author was supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant Am 60/9–2.

two circuits are isomorphic than to find out whether they are identical. Note the analogy with the same notion for graphs: two graphs on the same set of nodes are isomorphic if and only if they are identical after a permutation of the nodes of one of the graphs.

Another—though less intuitive—way of identifying Boolean functions is the following. Say that two Boolean functions are *negation equivalent* if one can negate some of the variables in one of the two functions such that the resulting Boolean function is identical to the other. For example, the two Boolean functions given by $x \wedge y$ and $\neg x \wedge y$ are negation equivalent (by negating x).

These two concepts can be combined: say that two Boolean functions are *congruent* if they are identical after a permutation of the variables and an additional negation of some of the variables. For example, the two Boolean functions given by $x \wedge (y \vee z)$ and $(x \vee y) \wedge \neg z$ are congruent. This equivalence relation, which already received attention in the last century, can be interpreted geometrically as a congruence of the two corresponding Boolean cubes, see Section 2.

Isomorphisms are defined by permutations. However, permutations are a special kind of bijective linear mappings on the $\text{GF}(2)$ vector space $\{0, 1\}^n$, namely, the ones whose matrices have exactly one 1 in each line and each row. So it is natural to consider the following generalization of the Boolean isomorphism relation: say that two Boolean functions $F(x_1, \dots, x_n)$, $G(x_1, \dots, x_n)$ are *linear equivalent* if there is a bijective linear mapping $i: \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that $F = G \circ i$. This relation is generalized to *Boolean affine equivalence* (and to *Boolean cardinality equivalence*) by bijective affine mappings (arbitrary bijective mappings) instead of bijective linear ones.

This paper states some results about the computational complexity of recognizing the above relations if the Boolean functions are represented as circuits (or formulas). For example, the computational problem corresponding to the congruence relation is the set of all pairs $\langle f, g \rangle$ such that f and g are circuits and the Boolean functions given by f and g are congruent.

The results in terms of polynomial time many-one reducibility are the following: the relations are situated between co-NP and Σ_2^P , the only exception is cardinality equivalence which is complete for the class CP. Furthermore, the problem whether two circuits are equivalent is complete for co-NP. The negation equivalence problem is reducible to the isomorphism and the congruence problem which have the same many-one complexity. These two problems are reducible to the linear and the affine equivalence relation, which have the same many-one complexity. A graphical summary is given in Figure 4.

Agrawal and Thierauf [1] solved a problem posed in some earlier version of this paper and showed that none of these equivalence relations is Σ_2^P -complete unless the Polynomial Time Hierarchy collapses.

2. Definition of the Equivalence Relations

Let $\{0, 1\}$ be the set of the two *Boolean constants*. A *Boolean function* is a function $F: \{0, 1\}^n \rightarrow \{0, 1\}$ for some natural number $n \geq 0$. The number n is called the *arity* of F and F will be written as $F(x_1, \dots, x_n)$. Boolean functions of different arities are different from each other. The tuples from $\{0, 1\}^n$ are called *assignments*. We use the usual formula/circuit notation in order to describe Boolean functions, for example,

the formula $x_1 \wedge \neg x_2$ describes the Boolean function $F(x_1, x_2)$ with $F(0, 0) = 0$, $F(0, 1) = 0$, $F(1, 0) = 1$, $F(1, 1) = 0$.

Now the equivalence relations on Boolean functions mentioned in the Introduction will be formally defined and some basic properties of them will be stated.

Let $F(x_1, \dots, x_n)$ be a Boolean function and let i be a function $\{0, 1\}^n \rightarrow \{0, 1\}^n$. Obviously, $F \circ i$ is also a Boolean function, remember that $F \circ i$ is defined by $(F \circ i) \times (x_1, \dots, x_n) = F(i(x_1, \dots, x_n))$. In this paper we only consider bijective functions $i: \{0, 1\}^n \rightarrow \{0, 1\}^n$, and some natural subsets of the set of these bijective functions are defined. First consider the set of functions $i: \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that $i(x_1, \dots, x_n) = (\pi(x_1), \dots, \pi(x_n))$ and π is a permutation on the set $\{x_1, \dots, x_n\}$. We call these functions *variable permutations* or just *permutations*. Another type of bijective functions are the functions $i: \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that $i(x_1, \dots, x_n) = (f_1(x_1), \dots, f_n(x_n))$ and each $f_i: \{0, 1\} \rightarrow \{0, 1\}$ is either the identity function or the negation function $n(0) = 1$, $n(1) = 0$. These functions are called *neg-mappings*, the prefix *neg* stands for *negation*. The two concepts can be combined: Let a *neg-permutation* be a composition $j \circ i$ of a permutation i and a neg-mapping j . Consider the set $\{0, 1\}^n$ as a vector space over the two-element field $\text{GF}(2)$, addition is given by pointwise parity \oplus . Note that a permutation is a bijective linear function on $\{0, 1\}^n$ with the special property that in every row and every line of the representing matrix there is exactly one 1. Therefore, bijective linear functions are a generalization of permutations. Likewise, neg-permutations are a special case of bijective affine functions on $\{0, 1\}^n$, namely, the ones of the form $i(\vec{x}) = l(\vec{x}) \oplus \vec{c}$ such that its linear part l is represented by a matrix of the special form like above.

Definition 1. Two Boolean functions $F(x_1, \dots, x_n)$ and $G(x_1, \dots, x_n)$ are said to be *isomorphic (negation equivalent, congruent, linear equivalent, affine equivalent, cardinality equivalent)*, written $F \sim G$ ($F \equiv_{\text{neg}} G$, $F \cong G$, $F \equiv_{\text{lin}} G$, $F \equiv_{\text{aff}} G$, $F \equiv_{\text{card}} G$), if there is a permutation (neg-mapping, neg-permutation, bijective linear function, bijective affine function, bijective function) $i: \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that $F = G \circ i$.

In other words, two Boolean functions (on the same set of variables) are isomorphic if and only if they are identical modulo a permutation of the variables, they are negation equivalent if and only if they are identical modulo a negation of some variables, they are congruent if and only if they are identical modulo a permutation of the variables and an additional negation of some of them. They are linear (affine, cardinality) equivalent if they are identical after the application of a linear (affine, any) bijective function on the assignments. It is obvious that two Boolean functions of the same arity are cardinality equivalent if and only if they have the same number of satisfying assignments.

An affine function $\{0, 1\}^n \rightarrow \{0, 1\}^n$ can be represented as a list of replacements $(x_1 \mapsto i(x_1), \dots, x_n \mapsto i(x_n))$, where each $i(x_i)$ is of the form $x_{j_1} \oplus \dots \oplus x_{j_k}$ or $x_{j_1} \oplus \dots \oplus x_{j_k} \oplus 1$. Note that the formulas $\alpha \oplus 1$ and $\neg\alpha$ are equivalent for all α . For the more special operations this representation is even easier, for example, the list $(x_1 \mapsto \neg x_3, x_2 \mapsto x_1, x_3 \mapsto \neg x_2)$ describes in an obvious way a neg-permutation on $\{0, 1\}^3$. Table 1 summarizes these representations. Remember that all operations have to be bijective.

Example. Let F, G, H, I be 2-ary Boolean functions defined by $F(x, y) = x$,

Table 1. Typical replacements.

Relation	Operation	Typical replacement
\sim	Permutation	$x_i \mapsto x_j$
\equiv_{neg}	Neg-mapping	$x_i \mapsto x_i$ $x_i \mapsto \neg x_i$
\cong	Neg-permutation	$x_i \mapsto x_j$ $x_i \mapsto \neg x_j$
\equiv_{lin}	Bijjective linear function	$x_i \mapsto x_{j_1} \oplus \dots \oplus x_{j_k}$
\equiv_{aff}	Bijjective affine function	$x_i \mapsto x_{j_1} \oplus \dots \oplus x_{j_k}$ $x_i \mapsto x_{j_1} \oplus \dots \oplus x_{j_k} \oplus 1$

$G(x, y) = y$, $H(x, y) = \neg x$, and $I(x, y) = x \oplus y$, respectively. Note that all functions are different from each other. The functions F and G are isomorphic by the permutation $(x \mapsto y, y \mapsto x)$ and F and H are negation equivalent by the neg-mapping $(x \mapsto \neg x, y \mapsto y)$. Therefore, G and H are congruent by the neg-permutation $(x \mapsto \neg y, y \mapsto x)$. The linear function $(x \mapsto x \oplus y, y \mapsto y)$ shows that F and I are linear equivalent. Therefore, H and I are affine equivalent by the bijective affine function $(x \mapsto x \oplus y \oplus 1, y \mapsto y)$.

In Figure 1 the Boolean functions which depend on at most two variables x and y are grouped according to the five equivalence-relations \sim , \equiv_{neg} , \cong , \equiv_{lin} , and \equiv_{aff} . Note that \equiv_{aff} and \equiv_{card} are identical on the Boolean functions with at most two variables. Nevertheless \equiv_{aff} is a proper subrelation of \equiv_{card} : Let $F(x, y, z)$ be 0 exactly on the assignments $\{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ (note that this set is not an affine subspace) and let $G(x, y, z)$ be 0 exactly on the (affine) linear subspace $\{(0, 0, 0), (1, 0, 0), (0, 1, 0), (1, 1, 0)\}$. F and G are cardinality equivalent, but because an affine function maps an affine subspace to an affine subspace, there cannot be a bijective affine function i such that $F = G \circ i$.

\sim	\equiv_{neg}	\cong	\equiv_{lin}	\equiv_{aff}
0	0	0	0	0
$x \wedge y$	$x \wedge y$	$x \wedge y$	$x \wedge y$	$x \wedge y$
$x \wedge (\neg y)$	$x \wedge (\neg y)$	$x \wedge (\neg y)$	$x \wedge (\neg y)$	$x \wedge (\neg y)$
$(\neg x) \wedge y$	$(\neg x) \wedge y$	$(\neg x) \wedge y$	$(\neg x) \wedge y$	$(\neg x) \wedge y$
$(\neg x) \wedge (\neg y)$	$(\neg x) \wedge (\neg y)$	$(\neg x) \wedge (\neg y)$	$(\neg x) \wedge (\neg y)$	$(\neg x) \wedge (\neg y)$
x	x	x	x	x
y	$\neg x$	y	y	y
$\neg x$	y	$\neg x$	$x \oplus y$	$\neg x$
$\neg y$	$\neg y$	$\neg y$	$\neg x$	$\neg y$
$x \oplus y$	$x \oplus y$	$x \oplus y$	$\neg y$	$x \oplus y$
$\neg(x \oplus y)$	$\neg(x \oplus y)$	$\neg(x \oplus y)$	$\neg(x \oplus y)$	$\neg(x \oplus y)$
$x \vee y$	$x \vee y$	$x \vee y$	$x \vee y$	$x \vee y$
$x \vee (\neg y)$	$x \vee (\neg y)$	$x \vee (\neg y)$	$x \vee (\neg y)$	$x \vee (\neg y)$
$(\neg x) \vee y$	$(\neg x) \vee y$	$(\neg x) \vee y$	$(\neg x) \vee y$	$(\neg x) \vee y$
$(\neg x) \vee (\neg y)$	$(\neg x) \vee (\neg y)$	$(\neg x) \vee (\neg y)$	$(\neg x) \vee (\neg y)$	$(\neg x) \vee (\neg y)$
1	1	1	1	1

Fig. 1. Partitions induced by the equivalence relations.

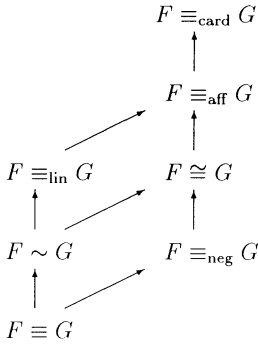


Fig. 2. Natural implications of the form $F \equiv_1 G \Rightarrow F \equiv_2 G$.

Proposition 2. *The relations $\sim, \equiv_{\text{neg}}, \cong, \equiv_{\text{lin}}, \equiv_{\text{aff}},$ and \equiv_{card} are equivalence relations. Figure 2 shows the inclusion relation among these equivalence relations.*

The equivalence relations above were already considered in the previous century, the relation of being congruent (in our terminology), especially, has received much attention since then, see, for example, [6], [8], [9], [10], [11], [16], [17]. The best overview about the definitions and results may be found in papers [9] and [10] by Harrison. It should be remarked that there does not seem to be a standard terminology, so we felt free to choose our own symbols and names. People studying these equivalence relations were not interested in the computational complexity of the relations. Instead, they were interested in determining the number and the size of the equivalence classes when only a fixed number of variables are involved, see Figure 1. The major breakthrough in that respect was achieved by Pólya in [16] who applied his famous general combinatorial result from [15] to the special case of congruence of Boolean functions.

Some Motivation for Boolean Congruence. Justifying its name, the Boolean congruence relation will easily be interpreted as a geometrical congruence problem, remember that two subsets of \mathbb{R}^n are called *congruent* if there is a distance-preserving function $\mathbb{R}^n \rightarrow \mathbb{R}^n$ which maps one subset bijectively to the other. Let a Boolean function $F(x_1, \dots, x_n)$ be given. The *geometrical Boolean cube representing F* is defined to be the subset of $\{0, 1\}^n \subset \mathbb{R}^n$ which consists of the tuples $t \in \{0, 1\}^n$ such that $F(t) = 1$. The Boolean congruence relation will also be interpreted as a graph isomorphism problem: Let the *graphical Boolean cube representing F* be the labeled undirected graph (N, E, λ) defined as follows. The set of nodes N consists of the 2^n different n -tuples from $\{0, 1\}^n$. The set of edges E consists of the pairs (t, t') of tuples which have Hamming distance 1, i.e., E is the set of (unordered) pairs $((c_1, \dots, c_n), (c'_1, \dots, c'_n))$ for which there is exactly one $i \in \{1, \dots, n\}$ such that $c_i \neq c'_i$ and $c_j = c'_j$ for all $j \neq i$. The labeling function $\lambda: N \rightarrow \{0, 1\}$ maps a tuple $t \in \{0, 1\}^n$ to the value of $F(t)$. See Figure 3 for this construction. The following proposition is proven in a straightforward way.

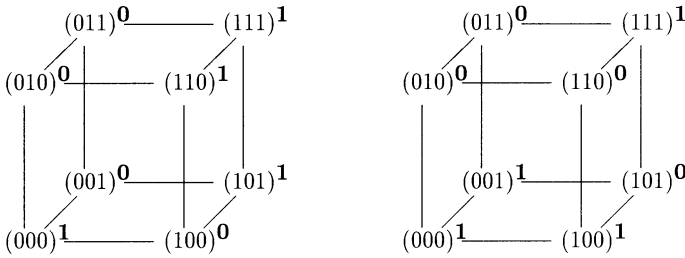


Fig. 3. Two (three-dimensional) graphical Boolean cubes which are isomorphic.

Proposition 3. For two n -ary Boolean functions F and G the following statements are equivalent:

- (a) F and G are congruent.
- (b) The geometrical Boolean cubes representing F and G are congruent.
- (c) The graphical Boolean cubes representing F and G are isomorphic.

3. The Complexity of the Equivalence Relations

This paper studies the complexity of the equivalence relations defined in the previous section when they are considered as computational problems. Karp [12] introduced the polynomial time many-one reducibility: A is reducible to B iff there is a polynomial time computable function h such that $w \in A \Leftrightarrow h(w) \in B$ for all w . Garey and Johnson [7] and Papadimitriou [14] give an overview on this and other standard notions from complexity theory. The notions p - m -reducible and p - m -equivalent are abbreviations for *polynomial time many-one reducible* and *polynomial time many-one equivalent*, respectively. The symbols for these relations will be as usual \leq_m^p and \equiv_m^p , respectively. The completeness and hardness notion will always refer to \leq_m^p .

We represent Boolean functions by circuits in order to obtain a computational problem. We could also use Boolean formulas but prefer the circuits since they are the more general and flexible concept while the results would be the same for both ways to represent the Boolean functions. The circuits use the constants 0, 1, the variables x_1, x_2, \dots , 1-ary negation (\neg), 2-ary conjunction (\wedge), disjunction (\vee), and parity (\oplus) gates, see, for example, [14] for a formal definition of circuits. We assume implicitly that a circuit is given as a pair $\langle n, c \rangle$ where n is a number denoting the intended arity of c , n has to be at least as large as the largest index i of a variable x_i appearing in the circuit c . This way, we can, for example, represent the n -ary constant-1 function by the pair $\langle n, 1 \rangle$. Nevertheless, in the context it will be always be clear what the intended arity of a circuit is, so we will, for example, still just write 1.

In the usual way each circuit $f(x_1, \dots, x_n)$ describes a Boolean function $\bar{f} = \bar{f}(x_1, \dots, x_n)$. Say that the circuits $f(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n)$ are *equivalent*, written $f \equiv g$, if they describe the same Boolean function, i.e., $\bar{f} = \bar{g}$, for example, $\neg(x_1 \vee x_2) \equiv \neg x_2 \wedge \neg x_1$. Let \mathcal{C} be the set of all encoded circuits and let $\langle \cdot \cdot \cdot \rangle$ be some

Table 2. Summarized terminology.

Relation name	Rel. symb.	Def. operation	Problem
Equivalence	\equiv	(Identity function)	BOOLE-EQUI
Isomorphism	\sim	Permutation	BOOLE-ISO
Negation equivalence	\equiv_{neg}	Neg-mapping	BOOLE-NEG
Congruence	\cong	Neg-permutation	BOOLE-CONG
Linear equivalence	\equiv_{lin}	Bijjective linear function	BOOLE-LIN
Affine equivalence	\equiv_{aff}	Bijjective affine function	BOOLE-AFF
Cardinality equivalence	\equiv_{card}	Any bijective function	BOOLE-CARD

usual pairing function on Σ^* . For any of the equivalence relations \equiv_r in Definition 1 we transfer the notion from the Boolean functions to the representing circuits, i.e., for two circuits f, g we write $f \equiv_r g$ if $\bar{f} \equiv_r \bar{g}$. The uniform definitions of the computational problems we consider are the following. Table 2 gives a summary of the terminology concerning the equivalence relations.

$$\begin{aligned} \text{BOOLE-EQUI} &= \{\langle f, g \rangle \mid f \equiv g\}, \\ \text{BOOLE-ISO} &= \{\langle f, g \rangle \mid f \sim g\}, \\ \text{BOOLE-NEG} &= \{\langle f, g \rangle \mid f \equiv_{\text{neg}} g\}, \\ \text{BOOLE-CONG} &= \{\langle f, g \rangle \mid f \cong g\}, \\ \text{BOOLE-LIN} &= \{\langle f, g \rangle \mid f \equiv_{\text{lin}} g\}, \\ \text{BOOLE-AFF} &= \{\langle f, g \rangle \mid f \equiv_{\text{aff}} g\}, \\ \text{BOOLE-CARD} &= \{\langle f, g \rangle \mid f \equiv_{\text{card}} g\}. \end{aligned}$$

The problems BOOLE-EQUI and BOOLE-CARD can be shown to be complete for well-known classes. The class CP was introduced in [19], the class is known to be in PSPACE and to include co-NP but it is neither known to include NP nor known to be in the Polynomial Hierachy.

Proposition 4.

- (a) BOOLE-EQUI is co-NP-complete.
- (b) BOOLE-CARD is CP-complete.

Proof. (a) The tautology problem $\text{TAUTOLOGY} = \{f \mid \forall t \in \{0, 1\}^n: f(t) = 1\}$ is known to be co-NP-complete. The problem BOOLE-EQUI is p-m-equivalent to TAUTOLOGY since $\langle f, g \rangle \in \text{BOOLE-EQUI} \Leftrightarrow \neg(f \oplus g) \in \text{TAUTOLOGY}$ and $f \in \text{TAUTOLOGY} \Leftrightarrow \langle f, 1 \rangle \in \text{BOOLE-EQUI}$.

(b) For a circuit $c(x_1, \dots, x_n)$ let $\#c$ denote the number of its satisfying assignments. The problem $A = \{f(x_1, \dots, x_n) \in \mathcal{C} \mid \#f = 2^{n-1}\}$ is known to be complete for CP. A is reducible to BOOLE-CARD by the reduction function which maps a circuit $f(x_1, \dots, x_n)$ to the pair $\langle f, x_1 \rangle$, note that $\#x_1 = 2^{n-1}$ (x_1 is considered as an n -ary function). On the other hand, BOOLE-CARD is reducible to A in the following way: given two circuits $f(x_1, \dots, x_n)$ and $g(x_1, \dots, x_n)$, consider the following circuit $h(x_1, \dots, x_n, x_{n+1})$:

$$(x_{n+1} \wedge f(x_1, \dots, x_n)) \vee (\neg x_{n+1} \wedge \neg g(x_1, \dots, x_n)).$$

Exactly $\#f$ tuples $(x_1, \dots, x_n, x_{n+1})$ satisfy $x_{n+1} \wedge f(x_1, \dots, x_n)$, and exactly $2^n - \#g$ of the tuples satisfy $\neg x_{n+1} \wedge \neg g(x_1, \dots, x_n)$. In total $\#f + 2^n - \#g$ of the tuples $(x_1, \dots, x_n, x_{n+1})$ satisfy h , in other words, $\#h = 2^n + (\#f - \#g)$. By this expression it is obvious that $\#f = \#g$ if and only if $\#h = 2^n$. Therefore, BOOLE-CARD is reducible to A by the reduction function which maps $\langle f, g \rangle$ to h . \square

Proposition 5. BOOLE-ISO, BOOLE-NEG, BOOLE-CONG, BOOLE-LIN, and BOOLE-AFF are co-NP-hard members of Σ_2^P .

Proof. The function $h \rightarrow \langle h, 1 \rangle$ is a many-one reduction from the tautology problem TAUTOLOGY to each BOOLE-ISO, BOOLE-NEG, BOOLE-CONG, BOOLE-LIN, and BOOLE-AFF. The reduction is verified by the observation that any bijective function i maps tautologies to tautologies, i.e., $f \in \text{TAUTOLOGY} \Leftrightarrow f \circ i \in \text{TAUTOLOGY} \Leftrightarrow \langle f, 1 \rangle \in \text{BOOLE-ISO}$ (BOOLE-NEG, BOOLE-CONG, BOOLE-LIN, and BOOLE-AFF, respectively).

Membership of the problems in Σ_2^P is witnessed by the following algorithm: for a circuit $f(x_1, \dots, x_n)$ first guess a representation list $(x_1 \mapsto i(x_1), \dots, x_n \mapsto i(x_n))$ representing a permutation (neg-mapping, neg-permutation, bijective linear function, bijective affine function) i , see Table 1. Then check for all assignments (x_1, \dots, x_n) from $\{0, 1\}^n$ whether $f(x_1, \dots, x_n)$ and $f(i(x_1), \dots, i(x_n))$ evaluate to the same value. \square

On the following pages the complexities of the problems BOOLE-ISO, BOOLE-NEG, BOOLE-CONG, BOOLE-LIN, and BOOLE-AFF will be compared with each other.

Theorem 6. BOOLE-ISO and BOOLE-CONG are p - m -equivalent.

Proof. One obtains a reduction from BOOLE-ISO to BOOLE-CONG as follows: Given two circuits f and g depending on x_1, x_2, \dots, x_n , the reduction constructs two new circuits c and d in the old variables x_1, x_2, \dots, x_n and the new additional variables y_1, y_2, y_3, y_4, y_5 :

$$\begin{aligned} c &= ((y_1 + y_2 + y_3 + y_4 + y_5 \geq 4) \wedge x_1 \wedge \dots \wedge x_n) \\ &\quad \vee (\neg y_1 \wedge \neg y_2 \wedge \neg y_3 \wedge f(x_1, \dots, x_n)), \\ d &= ((y_1 + y_2 + y_3 + y_4 + y_5 \geq 4) \wedge x_1 \wedge \dots \wedge x_n) \\ &\quad \vee (\neg y_1 \wedge \neg y_2 \wedge \neg y_3 \wedge g(x_1, \dots, x_n)). \end{aligned}$$

The disjunctive normal forms of both circuits contain exactly five monomials of degree $n + 4$, namely, the conjunctions of all variables x_1, x_2, \dots, x_n and four of the variables y_1, y_2, y_3, y_4, y_5 . Since all variables in these monomials appear in positive form, every neg-permutation witnessing $c \cong d$ has to preserve all variables in the positive form and thus is already a permutation of the variables. Therefore $\langle c, d \rangle \in \text{BOOLE-CONG} \Leftrightarrow \langle c, d \rangle \in \text{BOOLE-ISO}$.

Furthermore, y_1, y_2, y_3, y_4, y_5 belong to exactly four monomials of degree $n + 4$ while x_1, x_2, \dots, x_n belong to all five monomials of degree $n + 4$. So it follows that each x_k has to be mapped to some other x_m and any permutation witnessing $c \sim d$ already witnesses $f \sim g$: to see this fix the values of y_1, y_2, y_3, y_4, y_5 to 0 and the functions

of c and d thus restricted are just f and g . Therefore $\langle f, g \rangle \in \text{BOOLE-ISO} \Leftrightarrow \langle c, d \rangle \in \text{BOOLE-CONG}$.

A reduction from BOOLE-CONG to BOOLE-ISO is given the following way. Let a pair of circuits $\langle f(x_1, \dots, x_n), g(x_1, \dots, x_n) \rangle$ be given. Let y_1, \dots, y_n and z denote $n + 1$ new variables and define c, d by

$$\begin{aligned} c &= (x_1 \oplus y_1) \wedge \dots \wedge (x_n \oplus y_n) \wedge (z \vee f(x_1, \dots, x_n)), \\ d &= (x_1 \oplus y_1) \wedge \dots \wedge (x_n \oplus y_n) \wedge (z \vee g(x_1, \dots, x_n)). \end{aligned}$$

Now it is shown that $f \cong g \Leftrightarrow c \sim d$. If $f \cong g$ via a neg-permutation i , then $c \sim d$ via the following j :

$$\begin{aligned} j(z) &= z, \\ j(x_m) &= x_k \quad \text{and} \quad j(y_m) = y_k \quad \text{if} \quad i(x_m) = x_k, \\ j(x_m) &= y_k \quad \text{and} \quad j(y_m) = x_k \quad \text{if} \quad i(x_m) = \neg x_k. \end{aligned}$$

So the main idea is that the y_k represent $\neg x_k$ and so the negation is removed by introducing a new variable. The form of c and d enforces that $c(x_1, \dots, x_n, y_1, \dots, y_n, z) = 1$ only if $x_k = \neg y_k$ for $k = 1, \dots, n$ and, on the other hand, $c(x_1, \dots, x_n, y_1, \dots, y_n, z) = f(x_1, \dots, x_n)$ if $z = 0$ and $x_k = \neg y_k$ for $k = 1, \dots, n$. It follows from these observations that $f \cong g \Rightarrow c \sim d$.

Conversely, assume that $c \sim d$ via j . Call two variables v and w c -incompatible iff there is no satisfying assignment for c with $v = 1$ and $w = 1$. One can see that v and w are c -incompatible iff $v = x_k$ and $w = y_k$ for some k ; the same holds for the corresponding notion of d -incompatibility. If $c \sim d$ via j , then j has to map any pair of c -incompatible variables to a pair of d -incompatible variables: so for each k there is an m such that either $j(x_k) = x_m$ and $j(y_k) = y_m$ or $j(x_k) = y_m$ and $j(y_k) = x_m$. So one immediately obtains the neg-permutation

$$i(x_k) = \begin{cases} x_m & \text{if } j(x_k) = x_m; \\ \neg x_m & \text{if } j(x_k) = y_m. \end{cases}$$

It follows from $f(x_1, \dots, x_n) = c(x_1, \dots, x_n, \neg x_1, \dots, \neg x_n, 0)$ and the corresponding equality for g versus d that i witnesses $f \cong g$. \square

Theorem 7. BOOLE-NEG is p - m -reducible to BOOLE-ISO.

Proof. The following p - m -reduction from BOOLE-NEG to BOOLE-ISO is similar to the one from BOOLE-CONG to BOOLE-ISO. Let a pair of circuits $\langle f(x_1, \dots, x_n), g(x_1, \dots, x_n) \rangle$ be given. Choose $3n + 1$ different variables $y_1, y'_1, z_1, \dots, y_n, y'_n, z_n, z$ and construct the pair of circuits $\langle c, d \rangle$ where

$$\begin{aligned} c &= (\neg z_1)(y_1 \vee y'_1) \vee z_1(\neg z_2)(y_2 \vee y'_2) \vee \dots \vee z_1 z_2 \dots (\neg z_n)(y_n \vee y'_n) \\ &\quad \vee z_1 z_2 \dots z_n (y_1 \oplus y'_1) \wedge \dots \wedge (y_n \oplus y'_n) \wedge (z \vee f(y_1, \dots, y_n)) \end{aligned}$$

and d similarly depends on g . It holds that $f \equiv_{\text{neg}} g \Leftrightarrow c \sim d$. Verification is done the same way as for the reduction from BOOLE-CONG to BOOLE-ISO, where here the variables z_i guarantee that y_i is mapped only to y_i or y'_i . \square

Consider $\{0, 1\}^n$ to be a GF(2) vector space. Let $[x_k]$ denote the linear subspace $\{(a_1, \dots, a_n) : (\forall h \neq k) [a_h = 0]\}$ and let $[x_m, x_k]$ be the subspace generated by $[x_m]$ and $[x_k]$ and so on. Furthermore, $\mathbf{u}[x_m, x_k]$ denotes the projection (a_m, a_k) on the given variables. Note that if i is a bijective linear mapping, then i maps subspaces to other linear subspaces of the same dimension and cardinality.

Theorem 8. BOOLE-AFF is p - m -equivalent to BOOLE-LIN.

Proof. The m -reduction from BOOLE-AFF to BOOLE-LIN is $\langle f, g \rangle \rightarrow \langle x \wedge f, x \wedge g \rangle$ where x is a new variable. It remains to be shown that $\langle f, g \rangle \in \text{BOOLE-AFF}$ iff $\langle x \wedge f, x \wedge g \rangle \in \text{BOOLE-LIN}$:

$$f \equiv_{\text{aff}} g \Rightarrow x \wedge f \equiv_{\text{lin}} x \wedge g:$$

There is an affine mapping i witnessing $f = g \circ i$. Now let $j(x) = x$ and for variables y_m and z_m other than x one defines

$$j(y_m) = \begin{cases} z_1 \oplus \dots \oplus z_k & \text{if } i(y_m) = z_1 \oplus \dots \oplus z_k; \\ x \oplus z_1 \oplus \dots \oplus z_k & \text{if } i(y_m) = 1 \oplus z_1 \oplus \dots \oplus z_k. \end{cases}$$

For $x = 1$, $i(y_m) = j(y_m)$ and therefore $(x \wedge f)(1, y_1, \dots, y_n) = f(y_1, \dots, y_n) = g(i(y_1), \dots, i(y_n)) = (x \wedge g)(1, j(y_1), \dots, j(y_n))$. For $x = 0$, $(x \wedge f)(0, y_1, \dots, y_n) = 0 = (x \wedge g)(0, j(y_1), \dots, j(y_n))$ holds independently of the values $j(y_1), \dots, j(y_n)$. Thus $(x \wedge f) \circ j = x \wedge g$.

Since i^{-1} can be transformed in the same way to j^{-1} , the new linear mapping j is invertible and $x \wedge f \equiv_{\text{aff}} x \wedge g$.

$$x \wedge f \equiv_{\text{lin}} x \wedge g \Rightarrow f \equiv_{\text{aff}} g:$$

Let j be a linear mapping witnessing $x \wedge f \equiv_{\text{lin}} x \wedge g$. An affine subspace of a linear space is obtained from some linear subspace by adding a constant vector to every vector in this subspace. In particular, for Boolean vector spaces, any affine subspace is the closure of some set under the operation $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \rightarrow \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \mathbf{v}_3$. Now consider the affine subspaces V generated by the set $\{\mathbf{v} : (x \wedge f)(\mathbf{v}) = 1\}$ and W generated by the set $\{\mathbf{v} : (x \wedge g)(\mathbf{v}) = 1\}$. Since the linear mapping j satisfies $(x \wedge f)(\mathbf{v}) = (x \wedge g)(j(\mathbf{v}))$, j maps the vectors generating V to those generated W and so is an affine mapping between the subspaces V and W : $j(V) = W$. In particular, V and W have the same dimension and are also both affine subspaces of the subspace of all space U of all vectors which map the variable x to 1. Now there is an affine bijective mapping $h: U \rightarrow U$ which coincides with j on the set V of vectors. Now h can be made to an affine bijection i from the domain of f to that one of g as follows where y_m and z_m denote variables other than x :

$$i(y_m) = \begin{cases} c \oplus z_1 \oplus \dots \oplus z_k & \text{if } h(y_m) = z_1 \oplus \dots \oplus z_k \oplus c; \\ \bar{c} \oplus z_1 \oplus \dots \oplus z_k & \text{if } h(y_m) = x \oplus z_1 \oplus \dots \oplus z_k \oplus c. \end{cases}$$

Here, c is a Boolean constant (0 or 1) which comes from the affine mapping; linear ones do not have them.

This completes the proof that BOOLE-AFF is p - m -reducible to BOOLE-LIN. Conversely,

consider given functions f, g which have the variables x_1, \dots, x_n . Let y_1, \dots, y_{n+1} and z_1, \dots, z_{n+2} denote new variables and let $V \subseteq \{0, 1\}^{3n+3}$ be the vector space generated by all these $3n+3$ variables. Furthermore, let X be the n -dimensional Boolean subspace generated by the basis x_1, \dots, x_n , let Y be the $(n+1)$ -dimensional subspace generated by y_1, \dots, y_{n+1} , and let Z be the $(n+2)$ -dimensional subspace generated by z_1, \dots, z_{n+2} . So $\mathbf{v} \in X$ iff $\mathbf{v}(y_m) = 0$ and $\mathbf{v}(z_k) = 0$. Let $\mathbf{0}$ denote the shared 0-vector of all four vector spaces. Now the function $f: X \rightarrow \{0, 1\}$ is extended to a function $F: V \rightarrow \{0, 1\}$ as follows:

$$F(\alpha) = \begin{cases} f(\alpha) & \text{if } \alpha \in X; \\ f(\mathbf{0}) & \text{if } \alpha \in Y \cup Z; \\ \neg f(\mathbf{0}) & \text{otherwise, i.e., } \alpha \in W \text{ where } W = V - X \cup Y \cup Z. \end{cases}$$

Similarly, g is extended to G . Now it has to be shown that the mapping $\langle f, g \rangle \rightarrow \langle F, G \rangle$ is a p-m-reduction from BOOLE-LIN to BOOLE-AFF. Obviously the mapping is polynomial time computable.

$$f \equiv_{\text{lin}} g \Rightarrow F \equiv_{\text{aff}} G:$$

Let i witness that $f \equiv_{\text{lin}} g$. Since i is linear, $f(\mathbf{0}) = g(\mathbf{0})$. Given $\alpha \in X$, $\beta \in Y$ and $\gamma \in Z$, let $j(\alpha \oplus \beta \oplus \gamma) = i(\alpha) \oplus \beta \oplus \gamma$. Since i is linear, j must be affine. Since i is bijective, so is j . j maps X to X , therefore, for all $\alpha \in X$, $G(j(\alpha)) = G(i(\alpha)) = g(i(\alpha)) = f(\alpha) = F(\alpha)$. If $\alpha \in Y \cup Z$ then $j(\alpha) = \alpha$ and $F(\alpha) = f(\mathbf{0}) = g(\mathbf{0}) = G(\alpha)$. Since j maps $X \cup Y \cup Z$ to $X \cup Y \cup Z$, j also maps W to W . So for $\alpha \in W$, $F(\alpha) = \neg f(\mathbf{0})$ and $G(j(\alpha)) = \neg g(\mathbf{0}) = \neg f(\mathbf{0})$. The affine mapping j witnesses that $F \equiv_{\text{aff}} G$.

$$F \equiv_{\text{aff}} G \Rightarrow f \equiv_{\text{lin}} g:$$

Let i witness that $F \equiv_{\text{aff}} G$. Now F takes the value $f(\mathbf{0})$ only on $X \cup Y \cup Z$, that means on at most $2^n + 2^{n+1} + 2^{n+2}$ arguments while it takes the value $\neg f(\mathbf{0})$ at least on W . Since the cardinality of W is greater than that of $X \cup Y \cup Z$ but F and G map the same number of vectors to $f(\mathbf{0})$, it follows that $f(\mathbf{0}) = g(\mathbf{0})$.

Z is an $(n+2)$ -dimensional linear subspace where F takes the value $f(\mathbf{0})$. So G must on $i(Z)$ again take the value $f(\mathbf{0}) = g(\mathbf{0})$ and since $i(Z)$ is an affine $(n+2)$ -dimensional space, $i(Z) = Z$. Since Y intersects Z in one point, $i(Y)$ intersects $i(Z)$ also in one point. From this information it can be deduced that $i(Y) = Y$. Since $\mathbf{0} \in Y \cap Z$, $i(\mathbf{0}) \in i(Y) \cap i(Z)$ and it follows that $i(\mathbf{0}) = \mathbf{0}$, i.e., i is linear.

The set $U = X \cap i^{-1}(X)$ is a linear subspace of X . The restriction of i to U can be extended to a linear bijective function $j: X \rightarrow X$. Now assume by way of contradiction that $f(\alpha) \neq g(j(\alpha))$ for some $\alpha \in X$. Then $\alpha \notin U$. Thus $i(\alpha) \in W$ and $f(\alpha) = G(i(\alpha)) = \neg f(\mathbf{0})$. Further, there is γ with $i(\gamma) = j(\alpha)$. Since $j(\alpha) \notin i(U)$, $\gamma \in W$ and $g(j(\alpha)) = G(i(\gamma)) = F(\gamma) = \neg f(\mathbf{0})$. So such an α does not exist and j witnesses $f \equiv_{\text{lin}} g$.

Therefore, BOOLE-AFF is p-m-equivalent to BOOLE-LIN. □

Theorem 9. BOOLE-ISO is p - m -reducible to BOOLE-LIN.

Proof. Consider the circuits f, g with the variables x_1, \dots, x_n . The p - m -reduction $\langle f, g \rangle \rightarrow \langle c, d \rangle$ translates f to c and g to d by the formula below where x_1, \dots, x_n stand for the variables of f, g and $y_0, y_1, \dots, y_n, z_0, z_1, z_2$ are new additional variables only used in c and d . Now c assigns to every vector $\mathbf{u} \in [x_1, \dots, x_n, y_0, y_1, \dots, y_n, z_0, z_1, z_2]$ the following values:

$$c(\mathbf{u}) = \begin{cases} 0 & \text{if } \mathbf{u} \in [x_1, \dots, x_n, z_0, z_1, z_2] - [x_1, \dots, x_n] \\ & \text{or } \mathbf{u} \in [x_m, y_0, y_1, \dots, y_n]; \\ f(\mathbf{u}) & \text{if } \mathbf{u} \in [x_1, \dots, x_n]; \\ 1 & \text{otherwise.} \end{cases}$$

The definition for d is obtained by taking $g(\mathbf{u})$ instead of $f(\mathbf{u})$ in the second case of the formula above.

For verification first $f \sim g \Rightarrow c \equiv_{\text{in}} d$ is considered. Let i be an isomorphism witnessing $f \sim g$, then i can be extended to an isomorphism witnessing $c \sim d$ by defining $i(y_m) = y_m$ and $i(z_m) = z_m$ for all m . Since any isomorphism is also a linear mapping, it only remains to show that the mapping satisfies $c(\mathbf{u}) = d(i(\mathbf{u}))$ for all \mathbf{u} , which is left to the reader.

For the converse direction $c \equiv_{\text{in}} d \Rightarrow f \sim g$ let i be a linear bijection with $c(\mathbf{u}) = d(i(\mathbf{u}))$ for all $\mathbf{u} \in V$. The subspace $U = [x_1, \dots, x_n, z_1, z_2, z_3]$ is the only $(n+3)$ -dimensional subspace where c and d take at least $2^{n+3} - 2^n$ times the value 0. So U must be mapped onto itself: $i(U) = U$. The subspaces $U_m = [x_m, y_0, y_1, \dots, y_n]$ are the only $(n+2)$ -dimensional subspaces not contained in U on which c and d take at least $2^{n+2} - 2$ times the 0. So they have to be mapped onto each other, but may be permuted: $i(U_m) = U_{\sigma(m)}$ for some permutation σ . Since x_m is the only nonzero vector in $U_m \cap U$, its image $i(x_m)$ is the only nonzero vector in $U_{\sigma(m)} \cap U$, so $i(x_m) = x_{\sigma(m)}$. The restriction j of i to $[x_1, \dots, x_n]$ is a linear mapping which maps x_m to $x_{\sigma(m)}$. So j is generated by a permutation of the variables and $f \sim g$ via j . \square

Blass and Gurevich [2] defined the problem $\text{USAT} = \{c \in \mathcal{C} \mid c \text{ has exactly one satisfying assignment}\}$ and showed that it is co-NP-hard. Chang and Kadin [5] showed that USAT is not in co-NP unless the Polynomial Hierarchy collapses. The following construction is a p - m -reduction from USAT to BOOLE-NEG :

$$f(x_1, \dots, x_n) \rightarrow \langle f(x_1, \dots, x_n), x_1 \wedge \dots \wedge x_n \rangle.$$

Proposition 10. USAT is p - m -reducible to BOOLE-NEG .

Corollary 11. *If the Polynomial Hierarchy does not collapse, then BOOLE-NEG , BOOLE-ISO , BOOLE-CONG , BOOLE-LIN , and BOOLE-AFF are not in co-NP.*

The Graph Isomorphism problem GI [13] is the set of all pairs of graphs (here) with the same set of nodes such that there is a permutation i of nodes such that a pair (x, y) forms an edge in the first graph iff $(i(x), i(y))$ is an edge in the second graph; so GI

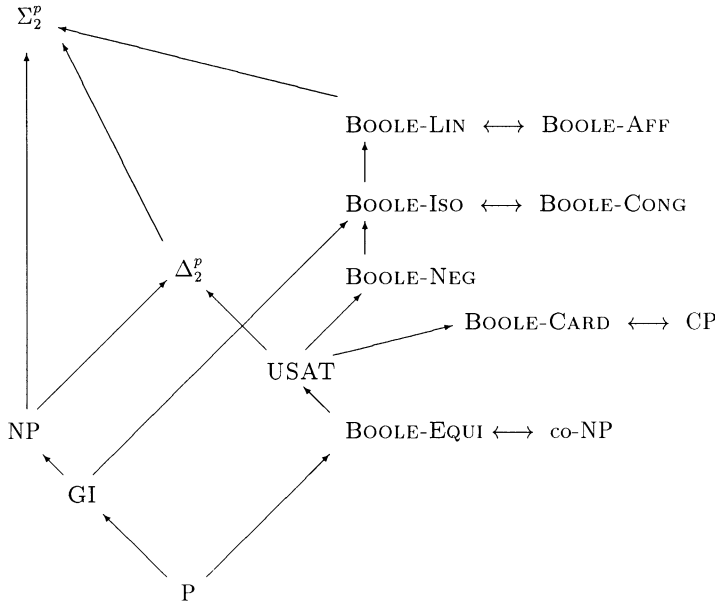


Fig. 4. Summary of the results.

is the graph analogue of BOOLE-ISO. Chang [4, Proposition 2] obtained the following result:

Proposition 12 [4]. *GI is p-m-reducible to BOOLE-ISO.*

Proof. Let for a graph $G = (V, E)$ the circuit h_G be defined as follows: for every vertex $i \in V$ in G choose a different variable v_i and let $h_G := \bigvee_{(i,j) \in E} (v_i \wedge v_j)$. Now, it is not difficult to see that G_1 and G_2 are isomorphic if and only if the two Boolean functions described by h_{G_1} and h_{G_2} are isomorphic. \square

Corollary 13. *BOOLE-ISO, BOOLE-CONG, BOOLE-LIN, and BOOLE-AFF are not in co-NP unless GI is in co-NP.*

The stated results are summarized in Figure 4 where an arrow denotes the proven existence of a p-m-reduction between two problems (in the case of a class consider a complete problem).

4. Related Work

In a previous version it was left open whether BOOLE-ISO and related classes are Σ_2^P complete. Agrawal and Thierauf [1] attacked this problem and showed that the complement

of BOOLE-AFF is in $\text{IP}[1]^{\text{NP}}$. Thus if BOOLE-AFF is Σ_2^p -complete then the Polynomial Time Hierarchy collapses. The same holds of course for the problems below BOOLE-AFF. So one of their main theorems is:

Theorem 14 [1]. *BOOLE-ISO is not Σ_2^p -complete unless the Polynomial Time Hierarchy collapses.*

This gives evidence to the conjecture that the problems considered here (besides BOOLE-CARD) have intermediate complexity between the first and the second level of the Polynomial Time Hierarchy, like Graph Isomorphism may have intermediate complexity between the bottom level and the first level of the Polynomial Time Hierarchy. The analogy between Graph Isomorphism and Boolean Isomorphism can be extended in the following way: Say that an equivalence relation e is induced by a preorder p if $e(x, y) \Leftrightarrow (p(x, y) \text{ and } p(y, x))$. For example, the equivalence relation \equiv_m^p is induced by the preorder \leq_m^p . Borchert and Ranjan [4] show that the equivalence relations \sim and \cong are induced by two preorders which express that one Boolean function is the (monotone) projection of the other, see [20]. Considered as computational problems on circuits these two preorders are Σ_2^p -complete, see [4]. Likewise, one step lower in the Polynomial Time Hierarchy, the graph isomorphism relation is induced by a preorder, namely, the subgraph isomorphism relation which as a computational problem is NP-complete, see p. 202 of [7].

Agrawal and Thierauf [1] improved the lower bound of Proposition 10 for the problems BOOLE-ISO, BOOLE-CONG, BOOLE-LIN, BOOLE-AFF by showing that the unique optimal clique problem UOCLIQUE is p-m-reducible to BOOLE-ISO. UOCLIQUE is a problem in $\text{P}^{\text{NP}[\log n]}$ which is not supposed to be complete for this class but which is still p-m-hard for USAT. In their paper they also study the automorphism problems which correspond to the equivalence relations defined in this paper. They show that similar results hold like in the case of Graph Isomorphism versus Graph Automorphism [13], e.g., the automorphism problems are p-m-reducible to the corresponding isomorphism problems.

Thierauf [18] considers Boolean functions presented by OBDDs and FBDDs. Since for these representations of Boolean functions the equivalence problems are in P and co-RP, respectively, the computational problems decrease (almost) one step in the Polynomial Time Hierarchy and are in NP and NP-co-RP, respectively.

Borchert et al. [3] took a closer look at the BOOLE-NEG problem and observed that it is a typical example of a problem being in the class EP^{NP} , where EP is the class of all sets computable via nondeterministic machines which have either no or 2^i for some i accepting paths. This property for BOOLE-NEG derives from the fact that for an instance of BOOLE-NEG the set of solutions is either empty or an affine subspace of $\{0, 1\}^n$.

Acknowledgments

The authors are grateful to Jin-yi Cai, Richard Chang, Antoni Lozano, and Thomas Thierauf for discussions about the subject.

References

- [1] M. Agrawal, T. Thierauf. The Boolean Isomorphism Problem, *Proceedings of the 37th Symposium on Foundations of Computer Science (FOCS)*, pp. 422–430, 1996; ECCC Report TR-96-032.
- [2] A. Blass, Y. Gurevich. On the unique satisfiability problem, *Information and Control* **55** (1982), 80–88.
- [3] B. Borchert, L. A. Hemaspaandra, J. Rothe. Powers-of-two acceptance suffices for equivalence and bounded ambiguity problems, submitted, 1996.
- [4] B. Borchert, D. Ranjan. The Subfunction Relations are Σ_2^P -Complete, Technical Report MPI-I-93-121, MPI, Saarbrücken, 1993.
- [5] R. Chang, J. Kadin. On the Structure of Uniquely Satisfiable Formulas, Technical Report 90-1124, Cornell University, 1990.
- [6] W. K. Clifford. On the Types of Compound Statement Involving Four Classes, *Memoirs of the Literary and Philosophical Society of Manchester*, Volume 16, No. 7, 1876–77, pp. 88–101. Reprint 1968: Chelsea, New York. Also in W. K. Clifford, *Mathematical Papers*, London, 1882, pp. 1–16.
- [7] M. R. Garey, D. S. Johnson. *Computers and Intractability*, Freeman, San Francisco, 1978.
- [8] M. A. Harrison. The number of transitivity set of Boolean functions, *Journal of SIAM* **11**(3) (1963), 806–828.
- [9] M. A. Harrison. On the classification of Boolean functions by the general linear and affine groups, *Journal of SIAM* **12**(2) (1964), 285–299
- [10] M. A. Harrison. Counting theorems and their applications to classification of switching functions, A. Mukhopadhyay, ed., in: *Recent Developments in Switching Theory*, Academic Press, New York, 1971, pp. 85–120.
- [11] W. S. Jevons. *The Principles of Sciences*, London, 1874 (in the second edition, London and New York, 1892, see pp. 134–146).
- [12] R. Karp. Reducibility among combinatorial problems, in: R. E. Miller and J. W. Thatcher, eds., *Complexity of Computer Computation*, Plenum, New York, 1972, pp. 85–103.
- [13] J. Köbler, U. Schöning, J. Toran. *The Graph Isomorphism Problem: Its Structural Complexity*, Birkhäuser Verlag, Basel, 1993.
- [14] C. Papadimitriou. *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.
- [15] G. Pólya. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen, *Acta Mathematica* **68** (1937), 145–254.
- [16] G. Pólya. Sur les types des propositions composées, *Journal of Symbolic Logic* **5**(3) (1940), 98–103.
- [17] C. E. Shannon. The synthesis of two terminal switching circuits, *Bell Systems Technical Journal* **28** (1949), 59–98 (see pp. 91–97).
- [18] T. Thierauf. The Computational Complexity of Equivalence and Isomorphism Problems, Habilitationsschrift, Fakultät für Informatik, Universität Ulm, 1997.
- [19] K. Wagner. The complexity of combinatorial problems with succinct input representation, *Acta Informatica* **23** (1986), 131–147.
- [20] I. Wegener. *The Complexity of Boolean Functions*, Teubner, Stuttgart, 1987.

Received July 1996, and in final form March 1998.