The Application of Polynomials over the Field of Two Elements to a problem in Intellectual Property

Gregory V. Bard*

October 26, 2008

1 Introduction

The application of verifying the equivalence of two digital circuits (i.e. that they output the same for all inputs) has been very well studied, and is usually accomplished using boolean algebra in some way, though there are many techniques. In this paper we will address a particular question in Intellectual Property, which is a generalization of determining the equivalence of two circuits, but yet is a sub-problem of the very difficult and less well-understood problem of "isomorphism of polynomials."

The author would like to thank an employee of the United States Department of Defense, who suggested this problem, but wishes to remain anonymous.

1.1 Background: Circuit Equivalence

This problem is crucial in the "formal verification" of digital circuits. The relationship to polynomials over $\mathbb{GF}(2)$, the field with two elements, arrises from the fact that a digital circuit is a collection of logic gates, and each logic gate is a simple polynomial over $\mathbb{GF}(2)$. Thus, because the composition of two polynomials is a polynomial, the entire circuit is a polynomial. For example, the following

Digital Gate	Logical Operation	Polynomial
x AND y	Conjunction	xy
x OR y	Disjunction	$x + y + xy$
NOT x	Negation	$1 + x$
x XOR y	Distinction	$x + y$
IF x THEN y	Implication	$1 + x + xy$
x NAND y	Incompatibility	$1 + xy$
x XNOR y	Equivalence	$1 + x + y$

are some simple logic gates. Thus converting a digital circuit to its polynomial system is straightforward, and results in one equation for each output bit, and one variable for each input bit. Let m denote the number of equations (number of outputs) and n the number of variables (number of inputs), as is standard.

The "circuit equivalence" problem consists of verifying if all outputs are equal for all inputs. This can be written

$$f_i(x_1, \ldots, x_n) + g_i(x_1, \ldots, x_n) = 1 \qquad \forall i \in \{1, 2, \ldots, m\}$$

because if that system of polynomial equations has a solution, then the two circuits are not equivalent. Likewise, if there is no solution, then the circuits are equivalent.

We will assume that an oracle exists to solve this problem. For example, one can write polynomials and use MAGMA's F4 and F5, or one can convert to circuit satisfiability and use a SAT-Solver. The principle measures of difficulty of the problem are the number of equations m and the number of variables n. If an algorithm solves this problem in time upper-bounded by a polynomial in m and n then P=NP.

*Department of Mathematics, Fordham University, The Bronx, NY, blah: add zipcode.

1.2 Background: The IP Problems

Consider some manufacturer A believes that manufacturer B has stolen and important circuit, and they wish to detect or confirm this condition. In reverse engineering circuit B, they might not know which output wires correspond to their own output wires, and likewise, which inputs of B correspond to which inputs of A. There are $m!$ possible permutations of the outputs and $n!$ possible permutations of the inputs, and thus a total of $n!m!$ possible permutations. Thus, one might naïvely believe that $n!m!$ oracle executions would be required. We will show that with some new variables and equations, this is not true.

If one views the polynomial system as a map $F : \mathbb{GF}(2)^n \rightarrow \mathbb{GF}(2)^m$, then this question can be modeled as follows. Does there exist an $n \times n$ permutation matrix P and an $m \times m$ permutation matrix Q such that

$$G(\vec{x}) = QF(P\vec{x}) \qquad \forall \vec{x} \in \mathbb{GF}(2)^n$$

We do not know if this has been studied before, but we name it "The Intellectual Property Problem", because if one drops the requirement that P and Q be permutation matrices, and require them instead to be merely non-singular, then one has the known and difficult "Isomorphism of Polynomials Problem" [blah: citation] The appendix contains a description of the connection between the Isomorphism of Polynomials Problem and public-key cryptography.

While algorithms for solving the "Isomorphism of Polynomials Problem" have been published for certain cases, and many of them are feasible, these algorithms are very complex and some forms of them have no known feasible countermeasure. The Intellectual Property Problem seems similar, so one might imagine it, too, is difficult.

2 The Solution

We will now solve the intellectual property problem, seeing if G is a copy of F merely with the inputs and outputs permuted. Consider for now $n = m = 8$. Then the polynomial function

$$\begin{aligned}
m(i_2, i_1, i_0, s_0, s_1, s_2, \ldots, s_7) &= (1+m_2)(1+m_1)(1+m_0)s_0 + (1+m_2)(1+m_1)(m_0)s_1 + \\
&= (1+m_2)(m_1)(1+m_0)s_2 + (1+m_2)(m_1)(m_0)s_3 + (m_2)(1+m_1)(1+m_0)s_4 + \\
&= (m_2)(1+m_1)(m_0)s_5 + (m_2)(m_1)(1+m_0)s_6 + (m_2)(m_1)(m_0)s_7
\end{aligned}$$

has the useful property that

$$m(i_2, i_1, i_0, s_0, s_1, s_2, \ldots, s_7) = s_{4i_2+2i_1+i_0}$$

and that in turn follows from the fact that this function will always have seven of its eight terms "zeroed-out" by i_2, i_1, i_0 regardless of the values that those three variables take. The remainder is the correct s. The circuit given by this polynomial is called an 8-way multiplexer.

We "attach" one of these to each input of G, in the following sense. Let

$$x'_j = m(i_{j,2}, i_{j,1}, i_{j,0}, x_0, x_1, x_2, \ldots, x_7)$$

where the x_js are the inputs to F, and m is the polynomial defined above. The x'_js will take the place of the x_js in the equations for G.

So long as the triple $(i_{j,2}, i_{j,1}, i_{j,0})$ is distinct from the triple $(i_{k,2}, i_{k,1}, i_{k,0})$ for all $j \neq k$, where $j \in \{1, 2, \ldots, 8\}$ and k likewise, then the mapping from the x_js to the x'_js is a permutation. This further can be caused to occur by adding the equation

$$\begin{aligned}
(i_{j,2}+i_{k,2}) + (i_{j,1}+i_{k,1}) + (i_{j,0}+i_{k,0}) + (i_{j,2}+i_{k,2})(i_{j,1}+i_{k,1}) \\
+(i_{j,2}+i_{k,2})(i_{j,0}+i_{k,0}) + (i_{j,1}+i_{k,1})(i_{j,0}+i_{k,0}) \\
+(i_{j,2}+i_{k,2})(i_{j,1}+i_{k,1})(i_{j,0}+i_{k,0}) &= 1
\end{aligned}$$

for all $k \neq j$ where $j \in \{1, 2, \ldots, 8\}$ and k likewise. This is because that equation will be satisfied in all cases unless $i_{j,\ell} = i_{k,\ell}$ for all $\ell \in \{0, 1, 2\}$.

The same operation can be performed on the outputs, converting y_js into y'_js and using $o_{j,2}, o_{j,1}, o_{j,0}$ in place of $i_{j,2}, i_{j,1}, i_{j,0}$.

The total cost of this is $3 \times 8 \times 2 = 48$ new variables, plus $2 \times 8 = 16$ new equations for the y' and x' definitions, and another $2\binom{8}{2} = 56$ equations for the guaranteeing of the non-identicality of the triples of is and of the triples of os. That is a total of 72 equations.

One oracle call is sufficient to answer the question after these changes, which is significantly less work than $(8!)^2 \approx 1.63 \times 10^9$ oracle calls.

Finally, note that this not only answers the decision question, but also explicitly identifies the specific permutations used on the inputs and the outputs, being the values of the $i_{1,0}, \ldots, i_{8,2}$ and the $o_{1,0}, \ldots, o_{8,2}$ variables, respectively.

2.1 The General Case

At the time of this writing, normally SAT-Solvers are able to handle 100s of variables, and 1000s of equations (for cryptographic problems[1]), and so while this is non-trivial, the system will remain feasible. This is much cheaper than

The $m \neq 8$ or $n \neq 8$ cases can be handled by allowing the second index of the is to go to $\lceil \log_2 n \rceil$ and of the os to go to $\lceil \log_2 m \rceil$. In this case there will be

$$n \lceil \log_2 n \rceil + m \lceil \log_2 m \rceil$$

new variables, and

$$m + n + \binom{n}{\lceil \log_2 n \rceil} + \binom{m}{\lceil \log_2 m \rceil}$$

new equations.

2.2 Very Large Cases

However, for the case $m = n = 32$, we would add 320 variables and 402,816 equations. This number of equations is not feasible by current methods. Omitting the equations that guarantee the non-identicality of the quintuplets seems foolish at first, but in reality, the following argument is likely to hold in any realistic scenario.

The equations that guarantee non-denticality of the quintuplets force P and Q as functions to be bijections. If two quintuplets are identical then two inputs of F map to the same input of G or two outputs of G map to the same output of F.

In the former case, by the pigeonhole principle, this means that an input of F was neglected, and not connected to G at all. If a solution to the entire system is found, that means that one input in the design of F was superfluous and not needed or used at all. This is very unlikely, but the prover knows about F and would be aware of that condition before beginning the problem. This case is therefore irrelevant to the practical problem.

In the latter case, again by the pigeonhole principle, this means that an output of G was matched with two distinct outputs of F, and in any solution of the system that means that these two outputs of Fs are always identical, otherwise how can they both equal a particular output of G simultaneously? This means an output of F was superfluous, merely a copy of another output of F. This can happen because of "fan-out" considerations, but again, the designers of F would know about this in advance, and one need not consider this case either.

[1]For easier problems, one may square these quantities, and the SAT-solver community has identified ensembles of very hard, very small problems.

Thus the equations for non-identicality can be dropped, and there will be only $m+n$ new equations. In this case, we are searching not only for P and Q that are permutation matrices, but the broader class of matrices with exactly one non-zero per row.

The only invertible members of this broader class of matrix are the permutation matrices themselves, since any matrix with two identical rows is singular. Therefore, this broader class is not of interest to the public-key system for which the Isomorphism of Polynomial problems was proposed.

A Application to Communications Security

The Isomorphism of Polynomials problem first was proposed in connection with communications security. Suppose one has c polynomial systems of equations each with n variables and n unknowns. Viewing these as a map $F : \mathbb{GF}(2)^{cn} \to \mathbb{GF}(2)^{cn}$ is possible, but it is far easier to find pre-images under this map F than some arbitrary G with the same domain and range. That is because the c polynomial systems could be solved separately. For example, compare factoring 40 numbers of 10 bits length, versus a 400-bit number.

Now fix two $cn \times cn$ dense invertible matrices over $\mathbb{GF}(2)$, call them S and T, and let $G(\vec{x}) = SF(T\vec{x})$. We must also assume that give G alone, it is hard to find S and T, and furthermore that it is hard to find other matrices that will likewise decompose G into a different, but still easy, system of equations. The system G will be the public key, and collectively T, S, F are the private key. To encrypt a message $\vec{x}$, one simply computes $G(\vec{x}) = \vec{c}$, which is easy, and transmits $\vec{c}$. The recipient calculates $v_1 = S^{-1}\vec{c}$, and then solves the c systems of n equations and n unknowns. Finally, one multiplies by T^{-1} to obtain the original message. An adversary would have to solve a system of cn equations in cn unknowns, which for careful parameter choices, is not feasible.