INF 240 - Exercise problems - 11

Nikolay Kaleyski

1 Boolean functions

A *Boolean function* on n variables is a function which maps a vector of n binary values (zeros and ones) to a single binary value. Formally, this is a mapping f from the n-dimension vector space $\mathbb{F}_2^n$ over the finite field $\mathbb{F}_2 = \{0, 1\}$ to $\mathbb{F}_2$.

Example 1. *Consider the parity function $p_3 : \mathbb{F}_2^3 \to \mathbb{F}_2$ on 3 variables, which outputs 0 if the number of ones that it receives as input is even, and outputs 1 if the number of ones in the input is odd. We then have $p_3(0,0,0) = 0$, $p_3(0,0,1) = 1$, $p_3(0,1,0) = 1$, $p_3(0,1,1) = 0$, $p_3(1,0,0) = 1$, $p_3(1,0,1) = 0$, $p_3(1,1,0) = 0$, and $p_3(1,1,1) = 1$.*

The simplest way to specify a Boolean function is by listing its output value for every possible input (which is what we have done in Example 1). When taking this approach, it is usually more systematic to list the values in a lookup table (such as the one in Table 1 below). This is called the *truth table (TT) representation* of f, and constitutes one of the fundamental methods of representing a Boolean function.

x_1	x_2	x_3	$p_3(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 1: Truth table of the parity function on 3 variables

When the number of variables n becomes large, however, the size of the truth table representation becomes restrictive. For instance, assuming that the truth table is represented optimally and requires a single bit per row, a Boolean function on $n = 40$ variables would require 2^{40} bits, which is precisely 128 gigabytes. For this reason, other representations of Boolean functions that may be more compact are considered.

One such representation is the *algebraic normal form (ANF)* of $f : \mathbb{F}_2^n \to \mathbb{F}_2$, which is nothing more than a polynomial over $\mathbb{F}_2$ in n indeterminates:

$$f(x_1, x_2, \dots, x_n) = a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + \dots + a_{2^n-1}x_1x_2 \dots x_n,$$

with the coefficients a_i being elements of $\mathbb{F}_2$ (hence either 0 or 1). The ANF may potentially contain all terms involving the n indeterminates $x_1, x_2, \dots x_n$, e.g. $x_1x_3x_7$, or x_2x_5, including the constant term 1 and the term $x_1x_2 \dots x_n$ containing all possible input variables. Thus, the ANF can have up to 2^n terms with non-zero coefficients; usually, the majority of the terms have zero coefficients, which leads to a more compact representation than the truth table.

In order to find the value of a function $f(x_1, x_2, \dots x_n)$ represented by its ANF, one simply substitutes concrete values for the indeterminates $x_1, x_2, \dots, x_n$ in the polynomial and simplifies the resulting expression over $\mathbb{F}_2$. For example, the ANF of the parity function p_3 on three variables is

$$p_3(x_1, x_2, x_3) = x_1 + x_2 + x_3.$$

Now, in order to find the value of p_3 at $(x_1, x_2, x_3) = (0, 1, 1)$, we simply substitute 0 for x_1 and 1 for x_2 and x_3 to get

$$p_3(0, 1, 1) = 0 + 1 + 1 = 0;$$

this is, of course, the same value as the one in Table 1.

Converting an ANF to a TT representation is as simple as evaluating the polynomial representing the function at every possible input and recording the values in a table.

Exercise 1. *Consider the Boolean function f on 4 variables given by the ANF*

$$f(x_1, x_2, x_3, x_4) = 1 + x_2 + x_1x_3 + x_1x_2x_4.$$

Find the truth table of f by filling in the blanks in Table 2.

x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Table 2: Truth table for $f(x_1, x_2, x_3, x_4) = 1 + x_2 + x_1x_3 + x_1x_2x_4$

An important statistic of the Boolean function that can be immediately extracted from ANF is its so-called algebraic degree. From the point of view of cryptographic applications, a high algebraic degree indicates a good resistance to higher order differential attacks. The *algebraic degree* of a Boolean function is the degree of its ANF, i.e. the number of indeterminates in the largest term with a non-zero coefficient. For example, in the function $f(x_1, x_2, x_3, x_4) = 1 + x_2 + x_1x_3 + x_1x_2x_4$ from Exercise 1, we have four terms with non-zero coefficients: 1 does not involve any indeterminates, x_2 has one indeterminate, x_1x_3 has two, and $x_1x_2x_4$ has three indeterminates. Thus, the algebraic degree of f is 3.

Functions of algebraic degree 1, resp. 2, resp. 3 are referred to as *affine*, resp. *quadratic*, resp. *cubic*. An affine function with no constant term is called *linear*.

We now look at how to convert a function from TT to ANF representation. One way of doing this is by means of so-called atomic functions. An *atomic function* is a Boolean function that evaluates to 1 for precisely one input (and hence evaluates to 0 for the remaining $2^n - 1$ inputs). If we are given a TT with, say, 5 zeros and 3 ones in the output column, and if we are able to construct the ANF's of the atomic functions corresponding to the three non-zero outputs, we can obtain the ANF corresponding to the TT by summing the ANF's for the atomic functions.

For example, consider the TT in Table 1. We already know that the ANF of this function is $p_3(x_1, x_2, x_3) = x_1 + x_2 + x_3$, but let us see how it can be extrapolated from the truth table using atomic functions. Looking at the TT, we see that p_3 evaluates to 1 for precisely four inputs, viz. $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$, and $(1, 1, 1)$. If we denote by f_1, f_2, f_4 and f_7 the corresponding atomic functions (with the index in the subscript being the integer whose binary expansion is the corresponding input vector), i.e. if

$$f_1(x_1, x_2, x_3) = \begin{cases} 1 & (x_1, x_2, x_3) = (0, 0, 1) \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x_1, x_2, x_3) = \begin{cases} 1 & (x_1, x_2, x_3) = (0, 1, 0) \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(x_1, x_2, x_3) = \begin{cases} 1 & (x_1, x_2, x_3) = (1, 0, 0) \\ 0 & \text{otherwise} \end{cases}$$

$$f_7(x_1, x_2, x_3) = \begin{cases} 1 & (x_1, x_2, x_3) = (1, 1, 1) \\ 0 & \text{otherwise} \end{cases}$$

and if we can find the ANF's of the above functions, then clearly $p_3 = f_1 + f_2 + f_4 + f_7$, so it is enough to take their sum.

Fortunately, there is a systematic way to construct the ANF of an atomic function. Consider the polynomial

$$(x_1 + 1)(x_2 + 1)x_3.$$

If $x_1 = 1$, then $(x_1 + 1) = (1 + 1) = 0$ and hence the entire polynomial evaluates to 0. So, in order for this polynomial to evaluate to 1, x_1 must necessarily be equal to 0. Similarly, the term (x_2+1) forces x_2 to be 0, and term term x_3 forces x_3 to be 1. In this way, this polynomial evaluates to 1 for $(x_1, x_2, x_3) = (0, 0, 1)$

and to 0 for all other combinations of inputs. In other words, this is precisely the ANF of the atomic function $f_1(x_1, x_2, x_3)$. Expanding the parentheses, we have

$$f_1(x_1, x_2, x_3) = x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3.$$

Similarly, we can construct the ANF of f_2 as

$$f_2(x_1, x_2, x_3) = (x_1 + 1)x_2(x_3 + 1) = x_2 + x_1x_2 + x_2x_3 + x_1x_2x_3,$$

of f_4 as

$$f_4(x_1, x_2, x_3) = x_1(x_2 + 1)(x_3 + 1) = x_1 + x_1x_2 + x_1x_3 + x_1x_2x_3,$$

and of f_7 as

$$f_7(x_1, x_2, x_3) = x_1x_2x_3.$$

The ANF of p_3 can then be computed as

$$p_3 = f_1 + f_2 + f_4 + f_7 = x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3 + x_2x_1x_2 + x_2x_3 + x_1x_2x_3 + \\ x_1 + x_1x_2 + x_1x_3 + x_1x_2x_3 + x_1x_2x_3 = x_1 + x_2 + x_3.$$

Recall that addition and subtraction coincide over $\mathbb{F}_2$ so that identical terms cancel out when simplifying the above expression.

Exercise 2. *Consider the Boolean function g on 4 variables given by the truth table*

x_1	x_2	x_3	x_4	$g(x_1, x_2, x_3, x_4)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Table 3: Truth table for $g : \mathbb{F}_2^4 \to \mathbb{F}_2$

Find the ANF's of the three atomic functions corresponding to the three non-zero outputs in Table 3, and use them to find the ANF of g. Determine the algebraic degree of g.

2 Vectorial Boolean functions

In general, the condition that the output of a Boolean function is a single bit is too restrictive for many practical applications. In cryptography, for instance, we would usually want to replace an input block of n bits with an output block of the same size – and not just a single bit. A natural way to implement multi-bit output using Boolean functions is to have several Boolean functions (instead of just one), with each function computing one individual bit of the output. For example, we could represent a function which maps 4 bits to 2 bits as a vector (f_1, f_2) of two functions, with $f_1 : \mathbb{F}_4 \to \mathbb{F}_2$ giving the first output bit and $f_2 : \mathbb{F}_2^4 \to \mathbb{F}_2$ giving the second output bit. Alternatively, we can consider a single function $F : \mathbb{F}_2^4 \to \mathbb{F}_2^2$. It is for this reason that functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ for some positive integers n and m are called *vectorial Boolean functions*, or (n, m)-functions. In particular, any Boolean function on n variables can be seen as an $(n, 1)$-vectorial Boolean function.

Vectorial Boolean functions can be represented using truth tables and polynomials in algebraic normal form just like Boolean functions, but the definitions of these two representations have to be slightly extended to accommodate for multiple output variables. In the case of the *truth table*, this simply means that multiple output columns are added. For example, the truth table of a $(3, 2)$-function is given below in Table 4.

x_1	x_2	x_3	$F(x_1, x_2, x_3)$	
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

Table 4: Truth table of a $(3, 2)$-function

If we consider F as a vector $F = (f_1, f_2)$ of Boolean functions, we say that f_1 and f_2 are the *coordinate functions* of F.

In the case of the *algebraic normal form* of an (n, m)-functions, the terms remain the same, but the coefficients are taken from $\mathbb{F}_2^m$ instead of just $\mathbb{F}_2$. The ANF corresponding to the function F given in Table 4 is

$$F(x_1, x_2, x_3) = (1, 1)x_1 + (1, 1)x_2 + (0, 1)x_3.$$

The ANF of an (n, m)-function can be obtained i.a. from the ANF's of its coordinate functions. Note that the second coordinate function, f_2, of F from Table 4 is precisely the parity function p_3 (compare with Table 1), and we already know that its ANF is $f_2(x_1, x_2, x_3) = x_1 + x_2 + x_3$. Upon inspecting the first output column of Table 4, we can see that f_1 gives the parity of the first two input columns, x_1 and x_2; thus, it is easy to see that its ANF is $f_1(x_1, x_2, x_3) = x_1 + x_2$. We can thus write

$$F(x_1, x_2, x_3) = (x_1 + x_2, x_1 + x_2 + x_3),$$

or, in ANF,

$$F(x_1, x_2, x_3) = (1,1)x_1 + (1,1)x_2 + (0,1)x_3.$$

Here, the coefficient in front of x_3 is $(0,1)$ since the term x_3 is in the ANF for f_2 but not in the ANF for f_1, while the coefficients for the remaining two terms are $(1,1)$ since these terms occur in the ANF's of both f_1 and f_2.

In this way, one can translate between the ANF and TT representation of vectorial Boolean functions through their coordinate functions.

Exercise 3. *Suppose that H is a $(3,4)$-function with coordinate functions*

$$\begin{aligned} h_1(x_1, x_2, x_3) &= 1 + x_1 + x_1x_2, \\ h_2(x_1, x_2, x_3) &= 1 + x_1x_2, \\ h_3(x_1, x_2, x_3) &= x_1x_2, \\ h_4(x_1, x_2, x_3) &= x_1 + x_1x_2x_3. \end{aligned}$$

Find the ANF of H and determine its algebraic degree. Derive the truth table representation of H by filling in the blanks in table 5.

x_1	x_2	x_3	$F(x_1, x_2, x_3)$
0	0	0	_ _ _ _
0	0	1	_ _ _ _
0	1	0	_ _ _ _
0	1	1	_ _ _ _
1	0	0	_ _ _ _
1	0	1	_ _ _ _
1	1	0	_ _ _ _
1	1	1	_ _ _ _

Table 5: Truth table for H

An important notion is that of the distance between (vectorial) Boolean functions: intuitively, distance is a measure of how "different" two given functions are. We typically use the notion of Hamming distance, which is simply the number of inputs for which two given functions produce a different output. In other words, given two (n,m)-functions F and G, the *Hamming distance* between F and G is defined as

$$d(F, G) = |\{(x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n \mid F(x_1, x_2, \ldots, x_n) \neq G(x_1, x_2, \ldots, x_n)\}|.$$

Thus, for instance, the Hamming distance between the two coordinate functions of $F(x_1, x_2, x_3)$ given in Table 4 is 4, since the two outputs bits are different for the inputs $(0,0,1)$, $(0,1,1)$, $(1,0,1)$, and $(1,1,1)$, and are equal for the remaining four inputs.

Exercise 4. *Consider the $(3,2)$-functions $F = (f_1, f_2)$ and $G = (g_1, g_2)$ given in Table 6. Find the Hamming distance between:*

1. *the two coordinate functions of F;*
2. *the two coordinate functions of G;*
3. *F and G.*

x_1	x_2	x_3	$F(x_1, x_2, x_3)$		$G(x_1, x_2, x_3)$	
0	0	0	0	0	0	1
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	1	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

Table 6: Truth table of $F = (f_1, f_2)$ and $G = (g_1, g_2)$

2.1 Univariate polynomial form

Recall that the elements of the finite field $\mathbb{F}_{2^n}$ can be seen as n-dimensional vectors of elements from $\mathbb{F}_2$; in other words, the finite field $\mathbb{F}_{2^n}$ and the vector space $\mathbb{F}_2^n$ can be seen as two representations of the same thing. When m divides n, this allows us to represent (n, m)-functions as univariate polynomials over $\mathbb{F}_{2^n}$. Since $x^{2^n} = x$ for any $x \in \mathbb{F}_{2^n}$, the degree of such polynomials can be assumed to be at most $2^n - 1$. The representation of an (n, m)-function F in the form

$$F(x) = \sum_{i=0}^{2^n-1} a_i x^i$$

for some coefficients $a_i \in \mathbb{F}_{2^n}$ is called the *univariate representation* of F. Although translating between the univariate and ANF or TT representation of a function is a bit more involved, one property of the function that can be immediately read off its univariate representation is the algebraic degree. The algebraic degree is equal to the maximum binary weight (number of ones in the binary representation) of any exponent in the univariate representation with a non-zero coefficient. For example, if

$$F(x) = x^{17} + \alpha x^7 + \alpha^{15} x^5 + \alpha^4$$

is the univariate form of a $(6, 6)$-function (where α is a primitive element of $\mathbb{F}_{2^4}$), we can see that the exponents with a non-zero coefficient are 17, 7, 5, and 0. Writing them in binary, we have $17 = 10001$, $7 = 111$, $5 = 101$, and $0 = 0$. Out of these, 7 has the largest binary weight (since it has 3 ones in its binary representation), so the algebraic degree of F is 3.

The degree of the polynomial representing the function in univariate form is called the *univariate degree*. In this case, the univariate degree is 17.

Exercise 5. *For the following* $(10, 10)$*-functions given in univariate form, compute their algebraic degree and their univariate degree. Are any of the functions linear, affine, quadratic, or cubic?*

1. $F(x) = x^{22} + x^{17} + \alpha x^6 + x^4 + 1$;
2. $G(x) = \alpha x^{15} + x^{11} + x^3$;
3. $H(x) = x^{256} + \alpha^{17} x^{64} + x^8 + x^4 + \alpha^2 x^2 + 1$.